(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 497 022 A1

EUROPEAN PATENT APPLICATION

(21) Application number: 91300772.0

(22) Date of filing: 31.01.91

(51) Int. Cl.5 G06F 15/40, G06F 9 44, H04M 3 56, H04N 7 14

(43) Date of publication of application: 05.08.92 Bulletin 92/32

(84) Designated Contracting States: DE FR GB

(71) Applicant: Hewlett-Packard Company
Mail Stop 20 B-O, 3000 Hanover Street
Palo Alto, California 94304(US)

(72) Inventor: Wink, Martin
c/o Hewlett-Packard Limited, Nine Mile Ride
Wokingham, Berkshire, RG11 3LL(GB)

Inventor: Jennings, Richard
c/o Hewlett-Packard Limited, Nine Mile Ride
Wokingham, Berkshire, RG11 3LL(GB)

Inventor: Baker, Colin
c/o Hewlett-Packard Limited, HP Labs, Filton Road
Stoke Gifford, Bristol, BS12 6QZ(GB)

(74) Representative: Smith, Denise Mary et al
Hewlett-Packard Limited Cain Road
Bracknell, Berkshire RG12 1HN(GB)

(54) Conference system.

(57) The present invention relates to a distributed object-based computer system in which sharable objects are split into client and server components (see Figure 7). Each client object contains a reference to the associated server object component. By copying client object components to other users, these other users obtain access to the relevant server-object component. This feature is described in the context of a distributed conferencing system

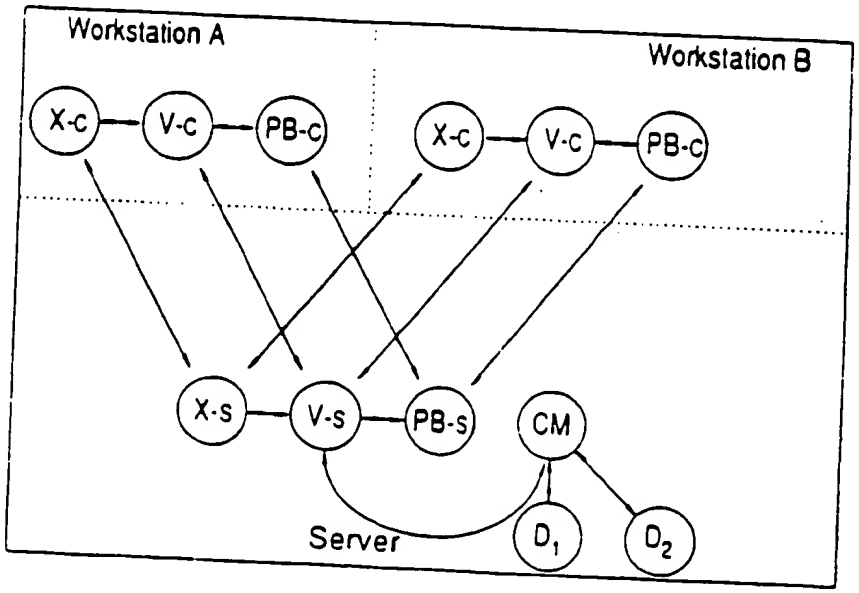


FIG 7

EP 0 497 022 A1

The present invention relates to a distributed computer system and relates particularly, but not exclusively, to a multimedia distributed object-based conference system.

The object-based approach to system development is becoming well-established. The basic idea is to program the system in terms of software objects, each having its own data and methods for operating on the data. Objects intercommunicate by means of messages. An advantage in encapsulating data and methods in this way is that the resulting system is relatively easy to maintain and develop. An example is NewWave Mail (produced and sold by Hewlett-Packard) which is an object-based electronic mail applications program in which messages and message components, such as text, distribution lists, etc. are treated as objects.

An object can be regarded as a discrete entity which can individually be moved, copied, destroyed, etc. An object is initially some data stored on disc or other medium. If object management software wishes to pass a message to it, one or more processes will be initiated which read the data as part of initialization. If an object is fully defined by its data and has no processes associated with it, it is said to be "inactive". If an object has one or more processes associated with it and is defined by the state of that process or processes and data then it is said to be "active".

A distributed object based system is one in which several workstations are interconnected over a network and messages between objects of the system can be sent over the network. Objects themselves may also be transmissible over the network. A network may comprise several interconnected intelligent workstations or a central computer connected to several terminals (workstations) or several interconnected server machines with intelligent workstations connected to each server, or a mixture of these possibilities. The term "workstation" is intended to be applicable to all of these possibilities.

In a distributed object based system there are benefits in splitting sharable semantic and presentation parts so as to enable more than one user to access the semantic part of a shared object. For example, in the context of a distributed conferencing system a whiteboard object would have a semantic part defining the state of the object and a presentation part for defining the appearance of the object to be displayed to a user and for enabling the user to make input. Several users may have access to a presentation part for viewing the whiteboard object so that they can each make contributions in a manner similar to a group of people clustered around a real whiteboard.

The workstations may be arranged in a client-server arrangement with semantic object parts stored on server machines and presentation object parts stored on client machines. Alternatively, semantic object parts may be distributed around user machines on a network of intelligent workstations.

According to the present invention we provide an object based distributed computer system comprising a network of workstations and means for transmitting objects between workstations characterized by objects including a first object type for storing data and a second object type for presenting data to a user, wherein objects of the second type reference an associated object of the first type to enable a plurality of users of workstations to access data of the object of the first type, comprising means for transmitting an object of the second type between workstations thereby to create a reference to the associated object of the first type for each workstation receiving an object of the second type.

The present invention provides an effective way of enabling further users to have access to a semantic object part, either for the purpose of autonomous working or for the purpose of participating in a joint activity.

In the embodiment to be described, the system comprises means for copying an object of the second type between workstations. In that embodiment transmitted objects of the second type include an identifier for the associated object of the first type.

The system according to the present invention may be in the form of a conferencing system comprising means enabling users of the workstations to participate in a meeting over the network wherein objects of the first type store meeting data and objects of the second type are for presenting meeting data. The invention also provides a method of convening a meeting using such a system comprising transmitting an object of the second type between workstations thereby to create a reference to the associated object of the first type for each workstation receiving an object of the second type.

It is believed that poor communications are a major cause of the poor performance of distributed teams of people working on a given project. The present invention advantageously provides an improved conference system for facilitating distributed meetings.

A particular embodiment of the present invention will now be described, by way of example, with reference to the accompanying drawings in which:

Figure 1 is a diagram of a distributed system according to the present invention;

Figure 2 shows the major components of a server and workstation of the system;

Figure 3 shows a voice and data network structure;

windows.

A possible video network is shown in Figure 5. The video network is based on a central video switch 54 connected using a star topology to client workstations C. Video signals are modulated on to VHF carriers and transmitted over standard analogue cabling 56. The video switch 54 is a conventional cable television switch. Several such switches can be cascaded in a bar arrangement for large systems.

For long distance video communications, a device 58 for compressing and decompressing video signals (a "codec") may be used and the signals are transmitted using ISDN telephone lines.

The architecture of the object-based system 10 will now be described.

With reference to Figure 6, the structure of one user's portion of the system is represented. The functions of the objects are as follows:

a Venue object (V) is an electronic meeting place allowing control over media channels and providing a location for storing shared objects. A user may have several Venue objects;

a Phone Booth object (PB) controls the creation of Venue objects and oversees the setting up, maintenance and closing down of conferences. The PB comprises a processor for handling incoming and outgoing calls;

a Connection Manager object (CM) controls driver components (D_0 ... D_n) which handle media connections for the system 10;

a Directory object (D) which provides a list of potential meeting participants.

Object X represents another system object for performing a specific meeting-related function, eg. a whiteboard function.

Figure 6 is a conceptual representation of the system 10 and the arrows represent inter-object communication. In the embodiment being described, the system comprises client workstations C and servers S and most of the objects referred to in Figure 6 are functionally-split into a server component and one or more client components as indicated in Figure 7.

The server objects handle the centralized and distribution - oriented aspects whereas the client objects handle the presentation aspects. Hence shared applications can be written with one server object connected to a plurality of client objects on different client workstations.

In Figure 7, PB-s means a Phone Booth server object and PB-c means a Phone Booth client object, and so on.

In this embodiment, the client objects are implemented as NewWave objects ie. several new classes of NewWave objects have been added: Venue objects, ROAM objects, Whiteboard objects, Phone Booth objects. Thus the semantic part of these functionally split objects runs on an HP-UX server and the user interface runs on MS-DOS NewWave client workstations.

The client workstations are each running an object-based system of the type described in European Patent Application No.339220A, the description of which is incorporated herein as Appendix A. Appendices A-D mentioned in attached Appendix A are not attached as part of this application but are incorporated herein by reference. Appendix A describes how objects are linked together by parent-child links and how objects can be copied. During a copy operation, the container of the object to be copied sends a message to the OMF28 asking the OMF28 to copy the relevant object and identifying the container object which is to receive the copy object.

The OMF28 performs the copy function and then sends a message to the target container object instructing it to insert the copy object as one of its child objects.

Mailing an object involves serialising the object, transmitting it to its destination and deserialising it. Serialising an object involves converting it to files, say DOS files, containing the data of the object and information about its properties and its child objects.

Server objects are not linked by parent-child links in the manner in which client objects are so linked. All client objects contain a reference to their associated server object. Figure 8 shows the form of data item 60 used to name objects. The data item 60 is an eight-byte array following the convention used for Internet Protocol (IP) addresses. The first 64 bits is a machine identifier M.I.D comprising a 32 bit server IP address and a 32 bit machine IP address. For a server object the server IP address and the machine IP address will be the same whereas for a client object these will be different. If there is only one domain per machine, the domain identifier D.I.D is zero. The object identifier O.I.D comprises a 32 bit generation count and a 16 bit tag. The 16 bit tag uniquely identifies the object within the relevant storage domain. Since tags are reusable when an object is deleted a generation count is used to ensure that each object is uniquely-named in time. The generation count is simply the time in seconds.

When a client object is closed (inactive) it appears as an icon on a user's screen. The user opens the object by clicking on the icon. Opening a client object causes it to send a message to its associated server object informing the server object that the client object is now active i.e. a Here Am I message. Until then

Figure 4 shows video facilities for a client workstation.
Figure 5 shows a video network structure:
Figure 6 illustrates the main objects in the system
Figure 7 illustrates the functionally split nature of the objects in the system.
Figure 8 shows the major components of the system infrastructure:
Figure 9 shows a typical Venue:
Figure 10 shows a CoMedian directory:
Figures 11 - 14 illustrate message sequences for system operations:
Figures 15 - 27 show screens during a typical user session.

The main components of a multi-media distributed object-based conferencing system according to the invention will first be described.

Referring to Figure 1, a multimedia distributed object-based conference system according to the present invention is indicated at 10. The system 10 comprises servers S connected over a network 12. The network 12 may be a wide area network (WAN) or a local area network (LAN) or a metropolitan area network (MAN). Client workstations C are connected to each of the servers S. Each site requires a server S.

Servers S communicate with each other by opening virtual circuits between pairs of servers. Although in principle, client workstations C could communicate directly with each other, this creates practical problems and therefore each client workstation C has only one virtual channel open to its local server S to enable client workstations to communicate with each other via servers S.

Referring to Figure 2, each server S comprises:

hardware 14, such as an HP9000 300 HP-UX computer (HP is a trade mark of Hewlett Packard Company);

operating system software 16, such as HP-UX software:

Remote Object Access Manager (ROAM) software 18 for managing communications with client workstations C connected to the server S and other servers on the network:

COM software 20 providing object management facilities:

server objects 21 which are objects to be shared between users and which correspond to the semantic object parts mentioned in the introduction.

Each client workstation C comprises:

hardware 22, such as an IBM-AT compatible PC:

operating system software 24, such as DOS software.

windowing software 26, such as MS Windows applications software:

an object management facility (OMF) 28, such as a Standard NewWave OMF. (Newwave is a trade mark of Hewlett-Packard Company used for a family of applications software):

objects software 30, such as NewWave objects and specialized client objects 32 and a ROAM object 34 for handling communication with objects on other computers. The client objects 32 correspond to the presentation object parts mentioned in the introduction.

The user of a client workstation C therefore has a windowed user interface within which to manipulate objects of the system and can cause objects to be transmitted over the network 12 via the associated server S.

The system 10 provides multimedia facilities to users. For example, each client workstation C may have voice and or video communication facilities as well as data communication facilities.

A possible voice and data network structure 40 is shown in Figure 3. In each of two sites designated A and B, a networked PC server 42 is connected to the local PABX. The PC server 42 contains one or more multi-port telephone interface cards (such as the VBX-300 card made by Natural Microsystems Inc.). The PABX is controlled by the PC server 42 and users can use their existing standard desk telephones 44 which are connected to the local PABX and conveniently located near their client workstations C.

Each of the sites A and B comprises a LAN and a LAN WAN bridge interconnecting the LAN with a WAN.

In use, the PC server 42 receives commands from servers S to set up, maintain and close down telephone conference calls. To the PABX, the PC server 42 appears as a normal telephone user and can therefore dial other users adding them in to conference calls using DTMF.

In order to conduct conferences over a wider area, PC servers 42a and 42b on respective sites A and B connect to each other over the public switched telephone network (PSTN) and add in their own local users to the conference.

Referring to Figure 4, each client workstation C with video facilities has a video camera 46, two or more VHF TV receivers 48, a microphone 50, a preamplifier 51 and a VHF modulator 52.

Furthermore, the client workstations C may be fitted with video cards to enable a user to view video in

Button Appearance	Meaning
No button	This person does not have this media channel available.
White, unhighlighted	The media channel is available, but not chosen for use.
Black	The media channel has been selected, but is inactive because the person is not present in the Venue or the connection has not been completed yet.
Red	The media channel is being used.

The lower portion of the Venue is taken up by the shared object area 78. This acts as a shared folder, storing objects on the server and making them accessible to all users of the Venue. Inactive objects are represented by an icon such as icon 88 in Figure 10. Objects in the shared object area 78 may be client objects e.g. Whiteboard client objects, or may be standard NewWave objects. It is possible to move objects into and out of the shared object area 78 of the Venue-client object. Moving a functionally-split object such as a Whiteboard object into the shared object area 78 does not entail moving the Whiteboard-server object but just the Whiteboard-client object. The OMF28 instructs the Venue client object to insert the Whiteboard-client object as one of its children. The Whiteboard-client object is then serialised by the Venue-client object and sent to the Venue-server object. The Venue-server object updates its other active Venue-client object with the news that a new Whiteboard object is available in the Venue and these Venue-client objects display the Whiteboard-client object icon in their shared object areas 78 accordingly. The Whiteboard-server object remains on whatever server it was initially stored. Subsequent opening of the Whiteboard object by any of the users of the Venue cause a copy of the Whiteboard-client object to be serialised by the Venue-server and sent to the relevant client-workstation where it is deserialised providing access to the contents of the Whiteboard object for that user. When that user subsequently closes (deactivates) the Whiteboard object, the copy of the Whiteboard-client object remains on that machine for subsequent use.

In contrast, if a NewWave object icon is moved into the shared object area 78 of a Venue-client object, this causes the NewWave object to be serialised and sent from the client workstation to the server machine which stores the relevant Venue-server object. The Venue-server object then instructs its other active Venue-client objects to display the relevant NewWave object icon. Subsequent opening of the Newwave object by a user of such an active Venue-client object causes a copy of the NewWave object to be made and sent to the relevant client workstation. Each such user thus obtains a separate copy of the NewWave object and changes which a user makes are not reflected in the copies held on the other users' machines. This is a consequence of the non-functionally split nature of NewWave objects and is an implementational feature rather than one which is important to the present invention.

There is one Phone Booth server object on every server machine and one Phone Booth client object on every client workstation. The Phone Booth client object arranges for the creation and activation of Venue client objects on client workstations and the Phone Booth server object manages the creation of Venue server objects and the convening of Venues. On opening a Phone Booth client object the user is presented with a directory 90 of possible meeting participants as shown in Figure 11. The directory 90 comprises a list 92 of potential participants, an area 94 for displaying a picture of a participant, a media selection area 96 and an options area 98 displaying three options: Convene Select and Cancel. Unavailable media options are greyed out in the area 96.

When a name is selected by choosing the Select option and then selecting a name from the directory 90, a picture of that participant appears in the area 94 as shown. The media connections are selectable by checking the relevant boxes in the media selection area 96. Checking the box beside the name of the person in the area 94 adds that person to the list of Venue participants. In addition, the initials of the media options chosen (Phone, Video, Data) appear against the participant's name in the list 92. A previously selected participant can be de-selected by de-checking the box beside the name of that person in the area 94. Taking the Cancel option means that none of the changes made since the window for the directory 90 was brought up will be implemented. The **Convene** option will be described later.

There is also a Connection Manager object on each server machine providing the facility to interconnect users using different media. The Connection Manager object handles the generic operations involved in establishing non-data interconnections. Drivers for each medium available, eg. video, telephone, handle the specific operations involved in carrying out switching requests during use. The Connection Manager object performs the following services:

maintains a list of media resources available in the system;

- detects when resources fail
- monitors resource channel availability (ie. monitors, microphones, speakers, cameras);

sets up connections between people using different media;

the server object is unaware of the existence of the client object. In other words, links between client and server objects are non-persistent and 'weak' i.e. the existence of a server object does not guarantee the existence of a corresponding client object and vice-versa. Server objects only store the identities of corresponding client objects which are currently active. Opening a client object means that a user can view the state of the object and can make input to it. The client object regularly updates, and is updated by, the server object.

Figure 9 depicts the components involved in a typical active server object which is associated with client objects on two different client workstations C_1 and C_2. Each object is given a unique object identifier comprising components identifying the relevant client server machine, the relevant storage domain and a number for the particular object. On the client side, the system has an object management facility (OMF) 60 for keeping a record of what objects are presently on the particular client workstation and which is involved in object creation and deletion, object naming, object activation and deactivation and inter-object message routing. This is a standard NewWave OMF. There is a client object manager library (COMLIB-C) 61 statically linked to each client object CO providing access to the functionality of a ROAM client object 62. In other words, the COMLIB-C 61 has been added to standard NewWave objects to form the client objects for functionally split objects. Communication through the COMLIB-C 61 is network transparent, ie. objects only need to know the object identifiers of other objects, not their locations.

On the server side there is a primitive object management facility (COM-S) 63 providing file management and object naming and message sending facilities in conjunction with the operating system software 64. A server object manager library (COMLIB-S) 65 is statically linked to each server object SO enabling access to the functionality of the object management facility 63 and a ROAM server object 66.

When client object CO_1 wishes to send a message to the corresponding server object SO, the ROAM client object 62 passes the message to the ROAM server object 66 which passes the message on to the server object SO. Messages from the server object SO to client objects are sent in the reverse manner. If a message is to be sent between objects on the same server the COMLIB-S 65 sends it directly without involving the ROAM server object 66. Messages are also sent between servers via the ROAM server object 66 and, in this way, communication between client workstations connected to different server machines is possible.

The functionality of certain objects in the system will now be described. The term "click" will be used in this specification to denote a selection made by the user of a workstation using an input device, such as a mouse. The term "drag" will denote moving the input device whilst such a selection is made so as to "drag" an item across the screen.

The Venue provides an electronic meeting room, inside of which person-to-person calls, group meetings and presentations to large groups can be held.

Venues provide a binding between the people involved in a meeting, the data which they are sharing, and the media channels connecting them. They are scalable from just two people up to many people, the exact number is subject to technical constraints. This allows a meeting to start off as a simple phone call between two people, build up as experts are brought in, to become a full group discussion without having to decide to move to a different object because the nature of the meeting has changed.

The Venue is a shared object and therefore exists on a server machine. The client workstations have Venue client objects which provide an interface to the Venue server objects running on the corresponding server object. There may be many Venue client objects on different client workstations for a particular Venue server object.

Figure 10 shows the appearance of a Venue to a user. The Venue is being viewed in a window 70 having a title bar 72 and a menu bar 74. At the top is a participants' area 76 where the people in the Venue can be seen and where their media channels can be controlled. Beneath that is a shared area 78 where objects for use in the meeting are stored.

The participants in a Venue are displayed side by side, with each participant being represented by a still bitmap 80, a name 82 accompanied by an indication of whether that user is present in the meeting or absent and status banner 84 indicating that an absent user has been invited to the meeting, and a row of media control buttons 86. The bitmap 80 may be replaced by a motion video window when video in windows is available and the video channel is in use.

Beneath the participants' area are three media buttons 86 for telephone, video and data and each one can be in one of four states. The states are:

workstation B to the Phone Booth server object PB-s. If the invitation is accepted a **Create Venue** message (referenced 6) is sent from the Phone Booth server object PB-s to the Phone Booth client object PC-c which causes it to create a new Venue client object V-c' on client workstation B involving sending a **Here Is Parent** message (referenced 7) to the new Venue - client object V-c' to notify it of the identity of the Venue server object V-s. The new Venue client object V-c' then sends a message (referenced 8) to the Venue server object V-s requesting information about the contents of the Venue. The reply from the Venue server object V-s is referenced 9 in Figure 13.

Messages corresponding to those referenced 6-9 are sent between the server S and client workstation A so as also to create a new Venue-client object V-c on that workstation and these messages are referenced 10-13 in Figure 13.

Finally, the Venue server object V-s sends a request (referenced 14) to the Connection Manager object CM to set up the chosen media connections and the Connection Manager object instructs the relevant media drivers accordingly (dotted line referenced 15).

The users of client workstations A and B can then communicate using the newly created Venue

It is also possible to convene an existing Venue by selecting the **Convene** option within the Venue. This initiates a sequence of events which will be described with reference to Figure 14. Again, a server machine S and two client workstations A and B are represented.

The user selection of the **Convene** option is referenced 1 in Figure 14. This causes the Venue client object V-c to send a **Convene Request** message (referenced 2) to the Venue server object V-s which notifies the Phone Booth server object PB-s of the convene request in a message referenced 3 which identifies the intended meeting participants. The Phone Booth server object PB-s sends a **Ring** message (referenced 4) to the Phone Booth client objects PB-c on the workstations of the intended meeting participants causing a dialogue box to be displayed on these workstations inviting the users to partake in a meeting. When these users accept or decline the invitation this causes a reply message (referenced 5) to be sent from each Phone booth client object PB-c' to the Phone Booth server object PB-s.

The next step is for the Phone Booth server object PB-s to instruct (message referenced 6) the Phone Booth client objects PB-c' to create new Venue client objects V-c' on machines where a Venue client object linked to the Venue server object V-s is not already stored. Such new Venue client objects V-c' then send a message (referenced 8) to the Venue server object V-s requesting information about the contents of the Venue so that the appropriate icons can be displayed in the shared area 78 of Figure 10 on the respective client workstations. The reply message containing information about the contents of the Venue from the Venue server object V-s is referenced 9 in Figure 13.

The Venue server object V-s then sends a request (referenced 10) to the Connection Manager object CM to set up the chosen media connections and the Connection Manager object instructs the relevant media drivers (not shown) accordingly (dotted line referenced 11). The distributed meeting can then proceed.

A user can also set up a new Venue by selecting a **Create a New** menu option in NewWave Office (Figures 14-17 of Appendix A). On opening the new Venue-client object a Venue-server object also needs to be created. Figure 15 depicts the process. A server machine is indicated by S and a client workstation by C.

The act of opening the new Venue-client object V-c causes it to send a message (referenced 1) to the Phone Booth client object PB-c which triggers a message (referenced 2) to be sent from the Phone Booth client object PB-c to the Phone Booth server object PB-s requesting creation of a new Venue server object V-s. The Phone Booth server object PB-s creates a new Venue server object V-s using a **Venue Start** message (referenced 3). Next the new Venue-server object V-s sends a **Here Is Parent** message (referenced 4) to the Venue-client object V-c containing the ID of the Venue-server object. The new Venue client object V-c then sends a message (referenced 5) to the Venue server object V-s requesting information about the contents of the Venue and there is a corresponding reply (referenced 6) from the Venue server object.

It is possible to add new meeting participants to an active Venue by selecting an **Add New Member** menu option. This causes a directory of potential participants to be displayed as shown in Figure 11 to enable the selection of one or more further participants and associated media connections. Information on these choices is conveyed from the Venue client object to the Venue server object which updates the control panels of the relevant Venue client objects. Chosen new meeting participants are not aware of any change until someone convenes a meeting.

When a user elects to close a Venue by selecting a **CLOSE** option this causes a message to be sent from the relevant Venue-client object to its Venue-server object informing the Venue-server object that the Venue-client object is deactivating. The Venue-server object then messages the Connection Manager object

- point-to-point
- multi-point: all that are available

maintains list of established connections and ensures synchronization with other networks ie maintains a model of the state of other networks.

optimizes switching to prevent unnecessary disconnect -connect transactions;

provides an interface for monitoring and auditing;

provides interface to media drivers.

Another functionally split object which is provided in this system is the Whitebcard. A Whiteboard object provides users with a shared computer whiteboard facility so that a user can draw, write and type on his her Whiteboard or acquire an image from another source and the input will be visible to other users viewing the same Whiteboard on different client workstations. Thus the Whiteboard object is an information sharing medium which allows users to look at a picture of what they are discussing.

Figure 12 shows an example of the appearance of a Whiteboard client object. The Whiteboard is being viewed in a window 100 having a title bar 102 and a menu bar 104. A drawing area 106 of the window 100 is devoted to displaying the contents of the Whitebcard. in this case a map showing the location of a Hewlett-Packard office. At the bottom of the window 100 is an area 108 indicating the range of tools which are available to the user of the Whiteboard. These tools comprise:

a scroller	110
a pointer	112
a selection of different coloured pens	114
an eraser	116
a text selector	118

Apart from the pointer 112. the tools are personal to a user ie each of the users viewing the same Whiteboard could be using the same tool eg. a red pen. without having to wait until another of the users had finished using that tool.

The scroller 110 can be used to scroll the entire window 100 around the Whiteboard. Selecting this tool turns the cursor into a compass enabling the view of the Whiteboard to be click-dragged around by the user.

Only one user can move the pointer 112 at a time. A user takes control of the pointer by clicking on the pointer logo - this turns the cursor into a pointer. At this time. the other users viewing the Whiteboard cannot see the pointer 112. To show the pointer 112. the user needs to click it down at a chosen point in the drawing area 106. The pointer 112 then becomes visible to all of the Whiteboard users at that chosen position. The cursor of the user who has just moved the pointer 112 reverts to the default arrow.

Likewise the seven coloured pens are selectable and deselectable by clicking on the appropriate pen logo. enabling different users to make input in different colours

The eraser 116 is selectable to remove marks on the Whiteboard. Also. direct typing of text onto the Whiteboard can be done by selecting the text selector 118.

In the area 108 there is also room for a status message 120 As users open or close the Whiteboard other users are notified by a status message.

Modes of operation of a system according to the present invention will now be described. concentrating first on utilization of the Venue.

Once a user selects participants and media as described with reference to Figure 11 and selects the Convene option a process of events is initiated to create a new Venue object. Figure 13 shows the objects and the numbered sequence of messages. Figure 13 depicts a server machine S and two client workstations A and B connected to the server machine S. On each client workstation there is initially a Phone Booth client object PB-c. On the server machine S there is initially a Phone Booth server object PB-s and a Connection Manager object CM.

On selecting the **Convene** option using client workstation A. which causes an input (dotted line referenced 1) to the Phone Booth client object PB-c. a message (referenced 2) is sent from the Phone Booth client object PB-c to the Phone Booth server object PB-s on the server machine S causing the Phone Booth server object to create a new Venue server object V-s using a **Venue Start** message (referenced 3). The Phone Booth server object PB-s then sends a **Ring** message (referenced 4) to the Phone Booth client object PB-c on client workstation B causing a dialogue box to appear on the screen of client workstation B inviting the user to take part in the proposed meeting That user accepts or declines the invitation causing a corresponding message (referenced 5) to be sent from the Phone Booth client object PB-c on client

To "open" an object, the user double clicks on the relevant icon. Referring to Figure 17, Martin has opened the Project Meeting Venue which is shown in a window 174. The window 174 has a menu bar 176 which has similar options to the menu bar 130 of the window 126 except a Meeting option instead of the Setting option. The window 174 displays a participants area 178, showing only Martin, and a shared items area 180 which is empty. Underneath a bit map 182 of Martin is a name bar 184 which includes a notification of presence and three media control buttons 185-7 for Phone, Video and Data respectively. Only the Data button 187 is highlighted in this example, ie, blacked out in Figure 17.

On selecting the Meeting option from the menu bar 176 of the window 174, a CoMedian directory window 190 appears, Figure 18. The reference numerals for the CoMedian directory which were used in Figure 11 will be retained here. Martin selects the name Richard Jennings from the list 92 of potential participants causing a picture of Richard to appear in the area 94 together with crosses in the video and data boxes in the area 96 to indicate Richard's media selections. This means that Richard will be contacted through the system for data sharing with both video and audio travelling over video connections. Martin then clicks on the Convene button in the options area 98 to add Richard to the Venue which causes Richard's image to join Martin's image in the Venue as shown at 192 in Figure 19. Richard is marked as absent at 194 and a banner 196 is displayed indicating that he has been invited. Martin has selected both video and data connections for himself in order to match what was selected for Richard. This causes the video and data buttons 186 and 187 to be highlighted in a first colour to show that they are currently in use albeit only locally to Martin's own workstation. Richard's video and data media buttons 186a and 187a are highlighted in a second colour to indicate that they have been requested but are not yet in use.

While waiting for Richard to join the Venue, Martin is moving the Design Notes and Design Principles objects 170 and 172 into the shared items area 180 of the Venue by clicking on each object and dragging it to the area 180.

Moving now to Richard's workstation, shown in Figure 20, the invitation to join the Venue has reached his machine and has caused a bell 200 to appear at the bottom of his screen. The bell 200 is flashing and making a ringing sound to attract his attention. Richard clicks on the bell 200 and the result is shown in Figure 21. An invitation message box 202 is brought up telling Richard that he has been invited to a meeting and giving the name of the meeting and the name of the person who convened the meeting. The invitation message box 202 comprises two options: Accept and Decline. Richard clicks on the Accept option to accept the invitation to join the meeting.

Referring to Figure 22 accepting the invitation causes a Venue client object automatically to be created and a window 204 to be opened for Richard. The chosen media connections have been set up so that Richard can now see and hear Martin and the objects that Martin has placed into the shared items area 180 are available to him. Figure 23 shows that Martin can see the same Venue having the same contents on his workstation. Referring to Figure 24, during the meeting, Martin has opened a window 206 on the Design Notes whiteboard object. Martin informs Richard of this so that Richard can also view the whiteboard object and then both Martin and Richard can scribble on the whiteboard and view each others input. When their meeting is finished both Martin and Richard close and save the Venue.

Figure 25 shows the Venue object 168 saved in Richard's NewWave office. In Figure 26, Richard has just opened his NewWave office and is viewing the Venue 168 in a window 208. Martin is not present (although he would be if, coincidently, he had his Venue open at the same time as Richard. In that situation, the relevant media connections would automatically be set up). Referring to Figure 27, Richard has selected the Meeting menu item using the cursor 210 so as to bring up the CoMedian directory 212 and he has selected Ed Davies in the manner previously described. Ed Davies does not have video capability, instead he is selected by telephone. Clicking on the Select button will cause Ed to be added to the Venue without beginning a Convene operation.

Referring to Figure 28, Richard is about to initiate a Convene operation by selecting the Action item from the menu bar 214 of the window 208, and selecting the Convene option from the corresponding menu 216. Since Ed does not have video capabilities, the audio from his telephone would be mixed into the video feed into Martin and Richard and their audio signals would be sent to Ed's telephone during their distributed meeting.

Turning now to Figure 29, a new session is beginning on Richard Jenning's workstation. A window 220 contains Richard's NewWave Office. Richard has created an outgoing message represented by the icon 222 called "Meeting Request" (using the "Create a New" option from the Action Menu - see Figures 14 to 17 of Appendix A). In Figure 30, on opening the outgoing message 222 it is displayed in a window 224. Richard has completed the distribution list 226 and written a cover note 228.

Referring to Figure 31, a new Venue-client object represented by the icon 230 is created (again using the "Create a New" option). The Venue-client object 230 is copied and dragged into the window 224

to disconnect the media connections for the Venue-client object which is deactivating. The Venue-server object sends messages to all of its other Venue-client objects informing them of the deactivation of the particular Venue-client object so that these other Venue-client objects alter their appearance to indicate that the relevant meeting member is now absent.

Another way of setting up a distributed meeting is for a user to copy an existing Venue-client object to the desired meeting participants. A Venue-client object is a reference to a Venue-server object. Copying a Venue-client object to other workstations creates a reference to the relevant Venue-server object on those other workstations because in the copying process the Venue-client object's reference to its Venue-server object is preserved.

There are different ways in which a Venue-client object can be copied to other workstations. One way is to include the Venue-client object in an electronic mail message. For this option, an electronic mail message is created in the normal manner e.g. using Hewlett-Packard's NewWave Mail and a Venue-client object is included in the message using a standard copy operation. When the or each addressee receives the message, they place the Venue-client in their collection of objects in preparation for the forthcoming meeting. At the relevant time, the meeting participants open their Venue-client objects to commence the meeting. On opening the Venue-client objects, their 32 bit machine IP address is automatically updated and the Venue-client objects send a Here Am I message to the associated Venue-server object.

Another option is for the user wishing to set up a distributed meeting to copy the relevant Venue-client object and to serialise the copy of the Venue-client object to a file on floppy disc (or other shared medium such as a network drive). This file may then be transported to the workstations of the intended meeting participants and deserialised thereby providing each of these participants with a copy of the Venue-client object and thereby means for accessing the associated Venue-server objects in order to take part in the distributed meeting.

A new Whiteboard-client object can also be created using the "Create A New" option in NewWave Office. On opening the Whiteboard-client object a new Whiteboard server object needs to be created. The process is analogous to that described with reference to Figure 15 replacing references to Venue objects with references to Whiteboard objects.

A new Whiteboard object can also be created inside a Venue by selecting the "Create a New" option inside the Venue. In this case, the Venue-client object automatically activates the new Whiteboard-client object in order to initiate creation of a new Whiteboard server object (again using a process analogous to that shown in Figure 15).

In the same manner as a Venue-client object can be copied and transmitted in an electronic mail message or via floppy disc, a Whiteboard-client object can be so utilised. Again the advantage of creating a reference to the relevant Whiteboard server object for the recipients of the copied Whiteboard-client objects is obtained since each copy of the Whiteboard-client object contains a reference to the Whiteboard server object (as described with reference to Figure 8).

Also as previously described, a Whiteboard-client object can be moved into the shared items area of a Venue object by a user causing copies of the Whiteboard-client object to be made available to the other users of the Venue object thereby giving access to the associated Whiteboard server object to these users.

An exemplary user session will now be described with reference to Figures 16 to 33 involving hypothetical users Martin, Richi and Ed.

Figure 16 shows a screen of a client workstation (Martin's) running Hewlett Packard NewWave Software. A window 126 has:

a title bar 128 carrying the title "NewWave Office"

a menu bar 130 offering the following options:

Action, Edit, Objects, View, Settings, Task and Help;

a system menu box 132;

size boxes 134 and 136;

a vertical scroll bar 138 with scroll arrows 140 and 142 and a scroll box 144;

a horizontal scroll bar 146 with scroll arrows 148 and 150 and a scroll box 152;

The window 126 displays icons for some standard tools at the top: Waste Basket 154, Agent 156, Printer 160, In Tray 162, Out Tray 164, File Drawer 166. The icons 168, 170 and 172 respectively on the left hand side represent work-related items:

"Project Meeting"	a Venue-client object representing a reference to a Venue server object on the local server machine;
"Design Notes"	a Whiteboard-client object representing a reference to a Whiteboard server object on the local server machine;
"Design Principles"	a NewWave document object fully contained on the client workstation

APPENDIX A

Brief Description of the Drawings

Figure 1 is a block diagram of a computer in accordance with the preferred embodiment of the present invention.

Figures 2 and 2A show block diagrams which illustrate the relationship between objects, applications and data files in accordance with the preferred embodiment of the present invention.

Figure 3 shows a plurality of objects linked in accordance with a preferred embodiment of the present invention.

displaying the message. This is achieved by clicking on the icon 230 and pressing the control key whilst dragging the icon into the message. (This is an alternative method from the user perspective to the copy procedure described with reference to Figures 18-20 of Appendix A.) The bar 232 labelled "Part 3" in Figure 32 shows that the message now contains a copy of the Venue-client object. The message window 224 is then closed (Figure 33). To send the message 222 it can be dragged onto the Out Tray icon 234. This causes a copy of the message, including the Venue-client object which it contains, to be sent to the people on the distribution list. The Out Tray object 234 initiates the serialisation of the message components to enable these to be transmitted over the network. On receipt at the respective destinations, the In Tray object represented by icon 236 deserialises the message components so that these can be viewed and manipulated by the recipients. The recipients can drag the Venue-client object out of the message and into their main NewWave Office window (220). At the appointed time, the three participants open their Venue-client objects to begin a distributed meeting. During the meeting, the users can open shared objects e.g. a Whiteboard object, and modify these interactively as well as interacting through their telephone and video interconnections. For example, input made by each user to a Whiteboard-client object is relayed to the Whiteboard server-object which updates all of the other corresponding active Whiteboard-client objects of the changes.

Although only Venue shared objects and Whiteboard shared objects are available to a user in this embodiment, it is envisaged that further possibilities for shared objects are a fax object, a discourse structurer object and tools to control the external media such as a virtual monitor manager and a video cassette recorder controller.

It is envisaged that a system according to the present invention may not entail the use of dedicated server machines but that server objects could run on user workstations given a suitable inter-object messaging infrastructure.

Figure 73 shows a block diagram of the organization of HPOMF.CAT, a system file included in the OMF shown in Figure 72.

Figure 74 shows the relation between a global parent and global objects in accordance with the preferred embodiment of the present invention.

Figure 75 is a block diagram which shows how system files within the OMF shown in Figure 72 accesses data files and applications from a memory shown in Figure 1.

Figure 76 is a block diagram of the organization of the memory shown in Figure 75.

Figure 77 and Figure 78 show objects and links in accordance with the preferred embodiment of the present invention.

Figure 79 is a block diagram of the organization of HPOMF.XRF, a system file included in the OMF shown in Figure 72.

Figure 80 shows a view specification record in accordance with the preferred embodiment of the present invention.

Figure 81 shows the use of a snapshot in accordance with a preferred embodiment of the present invention.

Figure 82 shows the data path of a view when there is no snapshot, in accordance with a preferred embodiment of the present invention.

Figure 4 shows a series of objects serving as folders, as parents of objects containing data, in accordance with a preferred embodiment of the present invention.

Figure 5 illustrates the screen display which results from linking of various objects in accordance with a preferred embodiment of the present invention.

Figure 6 shows the linking of objects in order to create the screen display shown in Figure 5.

Figure 7 shows how three objects may be linked together in accordance with a preferred embodiment of the present invention.

Figure 8 and Figure 9 illustrate how an object may be copied in accordance with a preferred embodiment of the present invention.

Figure 10 and Figure 11 illustrate the copying of a public object in accordance to a preferred embodiment of the present invention.

Figures 12 through Figure 71 show the appearance on a screen of a session in which a user manipulates objects in accordance with a preferred embodiment of the present invention. Also shown are block diagrams of how objects appearing to the user are linked in accordance to the preferred embodiment of the present invention.

Figure 72 is a block diagram of an Object Management Facility (OMF) in accordance with the preferred embodiment of the present invention.

18, OMF 100 informs the application which object the application should access for data. That object is then considered to be active. An object is inactive when the application the object is associated with is not being run by computer 18, or when the application the object is associated with is being run, but is not being run with the data of that object.

Active objects can communicate with each other using messages. For example if two instances of application 101 are being run by computer 18, one with the data of object 202 and the other with the data of object 203, object 202 and object 203 are both active. Therefore object 202 may send a message 211 to object 203. Similarly, if computer 18 is running application 101 with the data of object 202, and is running application 106 with the data of object 207, object 202 and object 207 are both active. Therefore, object 202 may send a message 212 to object 207.

Messages, such as message 211 and 212 may be formatted to be sent and received by all types of objects. This allows for free communication between all active objects. This also allows new object types to be defined and added to the system without requiring that the existing object types be updated to use the new type.

Each object has associated with it a set of data files. For instance, object 210 is shown to have associated with it a data file 221, a data file 222 and a data file 223. Data

Figure 83 shows the data path of a view when there is a snapshot, in accordance with a preferred embodiment of the present invention.

Description of the Preferred Embodiment

Figure 1 shows a computer 18 having a monitor 14, a keyboard 19 and a mouse 20. A portion of computer main memory 17 is shown by an arrow 9 to be within computer 18. Within computer memory main 17 is shown an object management facility (OMF)100, an application 101, an application 102, an application 103, an application 104, an application 105 and an application 106.

Each of applications 101 to 106 store data using objects. For instance, in Figure 2, application 101 is shown to have stored data using an object 202, an object 203, an object 204 and an object 205. Similarly, application 106 is shown to have stored data in an object 207, an object 208, an object 209 and an object 210. OMF 100 stores information indicating which objects go with which application. Objects which are associated with a single application are considered to be objects of the same type, or the same class. For instance, object 202, 203, 204 and 205 are of the same class because each is associated with application 101. Similarly objects 207, 208, 209 and 210 are of the same class because each is associated with application 106. All objects of the same class use the same application. When an application is being run by computer

children and one or more parents. An object is not allowed to become its own descendent.

In Figure 3 is shown an object 301, an object 302, an object 303, an object 304, an object 305, an object 306, an object 307, an object 308 and an object 309. Objects 301-309 have links with reference names which are numbers shown in parenthesis by each link. Object 301 has a link 310, with reference name "1", to object 302. Object 301 has a link 311, with reference name "2", to object 303. Object 302 has a link 312, with reference name "7", to object 304. Object 302 has a link 313, with reference name "8", to object 305. Object 303 has a link 314, with reference name "1", to object 306. Object 303 has a link 315, with reference name "4", to object 307. Object 304 has a link 316, with reference name "1", to object 308. Object 305 has a link 317, with reference name "7", to object 308. Object 306 has a link 318, with reference name "8", to object 309. Object 307 has a link 319, with reference name "9", to object 306. Object 307 has a link 320, with reference name "13", to object 309. Object 308 has a link 321, with reference name "1", to object 309. Object 308 has a link 322, with reference name "3", to object 303.

Object 301 is a parent of 302 and 303. Object 303 is a child of object 301 and of object 308. Each of objects 302-309 are descendents of object 301. Descendents of object 303 are objects 306, 307 and 309. Object 309 has for

in data files 221, 222 and 223 are in a format which can be interpreted by application 105.

Each object has associated with it a list of properties. Each property has a name and a value which may be accessed by specifying the name. In addition, each class of objects has associated with it a list of properties that are common to all objects of that class. For instance, in Figure 2A, object 205 and application 101 are shown. Object 205 has associated with it a property 231, a property 232, and a property 233. Application 101 has associated with it a property 131, a property 132 and a property 133.

Property lists can contain any number of properties. Each property value can be from zero to 3,2762 bytes in length. Properties are used to store descriptive information about objects and classes, such as names, comments and so on.

Objects may have references to other objects. These references are called links. Links are directional: one object is called the parent, the other the child. Each link has a reference name which is a number that is assigned by the parent object to identify each of its children. All of an object's children, its children's children, and so on are collectively called that object's descendents. Similarly, an object's parents, its parents' parents, and so on, are collectively called that object's ancestors. In the preferred embodiment of the present invention, an object which may be manipulated by a user, can have zero or more

contains lines of text 511, lines of text 512, a graphics figure 513, a graphics figure 514 and spreadsheet data 515. As shown in Figure 6, text and formatting data is stored in an object 611, graphics data for graphics figure 513 is stored in an object 612, graphics data for graphics figure 514 is stored in an object 613 and spreadsheet data 515 is stored in object 614. Links that are used to build compound objects always have some kind of data transfer associated with the link and hence are called data links. In Figure 6 is shown a data link 615, a data link 616 and a data link 617. In document 510, data from object 612, object 613 and object 614 are merely displayed, therefore data link 614, data link 615 and data link 616 are visual data links. In a visual data link, the parent will send requests to its child to display data within the parent's window.

In Figure 7, an object 701, which contains data for a first spreadsheet, is linked through data link 704 to an object 702, which contains data for a second spreadsheet, and is linked through data link 705 to an object 703, which contains data for a third spreadsheet. The first spreadsheet uses data from the second spreadsheet and from the third spreadsheet. Since the first spreadsheet does more than merely display data from the second and the third spreadsheets, data link 704 and data link 705 are called data-passing data links.

OMF 100 does the "bookkeeping" when objects are copied or mailed. When an object is copied, OMF 100 makes copies

ancestors all of objects 301-308. Object 303 has for ancestors objects 301, 302, 304, 305 and 308. And so on.

Active objects can dynamically make and delete links to other objects. When a link to an object is deleted, OMF 100 checks if the object has any other parents. If not, OMF 100 destroys the object by deleting the data files of the object and reclaiming other storage space associated with the object.

Object links may be used for various purposes. For example, folders may be in the form of objects. The children of objects used as folders may be objects containing data for use with various applications, or the objects may be other folders. Figure 4 shows an example of the use of objects as folders. An object 401 (also called folder 401), an object 402 (also called folder 402), an object 403 (also called folder 403) and an object 404 (also called folder 404) are used as folders. Folder 401 contains an object 405, used to contain data, an object 406, used to contain data, an object 407, used to contain data, and folder 402. Folder 402 contains an object 408, used to contain data, folder 403 and folder 404. Folder 403 contains an object 409, used to contain data, and an object 410, used to contain data. Folder 404 contains an object 411, used to contain data, an object 412, used to contain data and an object 413, used to contain data.

A more sophisticated use of links is to construct compound objects. For instance in Figure 5, a document 510

162 through a new link 163a. Object 161a is a copy of object 161. Link 163a is a copy of link 163.

In Figure 12 through Figure 71, it is shown how objects are displayed to a user on monitor 14. In Figure 12 a "NewWave Office" desktop is shown to include icons labelled as "File Drawer", "Waste Basket", "Diagnostic", "Printers", "Star" and "My Folder". A user (not shown) has manipulated a cursor 781, using keyboard 19 or mouse 20, to select "My Folder".

Figure 13 shows how the objects displayed on monitor 14 are linked. NewWave Office (shown as an object 700) is the parent of "File Drawer" (shown as an object 701) through a link 711, of "Waste Basket" (shown as an object 702) through a link 712, of "Diagnostic" (shown as an object 703) through a link 713, of "Printers" (shown as an object 704) through a link 714, of "My Folder" (shown as an object 705) through a link 715 and of "Star" (shown as an object 706) through a link 716.

In Figure 14, the user, using cursor 781, has selected "Create a New..." in a pull down menu 782. As a result of this selection a dialog box 779 appears as shown in Figure 15. Using cursor 781, the user has highlighted the icon "Layout" and using keyboard 19 has typed in the name "Paste Up" as a name for a new object to be created. Cursor 781 now points to a region labelled "OK". Once this region is selected, a new object titled "Paste Up" is created, as is shown in Figure 16.

of data files associated with the object. If the object being copied has children, OMF 100 also makes copies of the object's descendents, and builds links between the new objects to give the new compound object the same structure as the original.

For instance, Figure 8 shows object 308, from Figure 3, and the descendents of object 308. When OMF makes a copy of object 308, OMF copies each of object 308's descendents and the links shown in Figure 8. Figure 9 shows a copy of object 308. Object 308a is a copy of object 308. Object 303a is a copy of object 303. Object 306a is a copy of object 306. Object 307a is a copy of object 307. Object 309a is a copy of object 309. Link 321a is a copy of link 321. Link 322a is a copy of link 322. Link 314a is a copy of link 314. Link 315a is a copy of link 315. Link 318a is a copy of link 318. Link 319a is a copy of link 319. Link 320a is a copy of link 320.

In the preferred embodiment, the default behavior results in the copy of a parent's children when the parent is copied. However, when a child is designated as "public" it is not copied. Rather, a copy of the parent includes a link to the child. For instance, in Figure 10, a parent object 161 is to be copied. Parent object 161 is linked to a child object 162 through a link 163. Child object 162 is a public object. As shown in Figure 11, copying of parent object 161 results in new object 161a being linked to object

In Figure 23, using cursor 781, "Paste Up" (object 708) is shown being dragged to window 785. In Figure 24, the process is complete and "Paste Up" (object 708) is now in window "My Folder". In Figure 25, "Paste Up", shown as object 708, is now a child of "My Folder" through link 728.

The user sets up multiple links by using the Share command. This command is an extension of the clipboard metaphor common in software packages today for moving and copying data around the system. The clipboard is a special buffer that the system uses to hold data that is in transit.

In one way, the Share command operates similarly to the Cut or Copy command described above. That is, using Share, Cut, or Copy, the user selects some data first and then gives the Share command, which results in something being put on the clipboard. In the case of the Share command, however, what is put on the clipboard is neither the actual data nor a copy of the actual data. Instead, it is a link to the selected data. When this link is pasted, a permanent connection is made between the original data and the location of the Paste. Through use of OMF 100, this link is used by the involved applications to provide easy access to the original data (in its full application) and automatic updating when the original data is modified.

In Figure 26, the NewWave Office window has been activated. "Paste Up" (object 707) has been selected, as evidenced by "Paste Up" (object 707) being in inverse video. Using cursor 781, "Share" from menu 783 is selected. In

In Figure 17, "Paste Up" is shown as an object 707 linked as a child of NewWave Office through a link 717.

The basic clipboard operations are Cut, Copy, and Paste. The user must select the data that is to be moved or copied, and then give either the Cut command or the Copy command. Cut moves the selected data to the clipboard (deleting it from its original location). Copy makes a copy of the selected data on the clipboard. The user must then select the location where he wants the data to be moved or copied to, and give the Paste command. This command copies the contents of the clipboard to the selected location.

In Figure 18 a user is shown to have selected "Paste Up". The selection is represented by the icon for "Paste Up" being displayed using inverse video. With cursor 781, the user selects "Copy" from a pull down menu 783. In Figure 18A a Clipboard object 720 is shown to be a parent of an object 708 through a link 721. Object 708, is a copy of object 707 ("Paste Up").

As shown in Figure 19, next the user selects "Paste" from pull down menu 783. The result, shown in Figure 20, is the addition of an object 708, pointed to by cursor 781, which is a copy of the original "Paste Up" object 707.

In Figure 21, the new object is shown as object 708 linked as a child of NewWave Office through a link 718.

In Figure 22, "My Folder", has been opened by double clicking the icon for "My Folder" using cursor 781. The result is a new window 785 representing "My Folder".

spreadsheet could cause a graph to be re-drawn, and updated as a figure in a document. And since an object can have many parents, a single object can be used as "boiler plate" for any number of other objects. A change in the boiler plate will be reflected in all the objects which have links to it. Automated data transfer is illustrated in the following discussion.

In Figure 30, window 785 for "My Folder" has been closed. In Figure 31, cursor 781 is used to select "Create a New..." from pull down menu 782. As a result of this selection dialog box 779 appears as shown in Figure 32. Using cursor 781, the icon HPText has been highlighted and using keyboard 19 the name "Sample Text" has been typed in as the name for a new object to be created. Cursor 781 now points to a region labelled "OK". Once this region is selected, a new object titled "Sample Text" is created, as is shown in Figure 33.

In Figure 34, "Sample Text" (object 709) is shown to be a child of NewWave Office through a link 719. In Figure 34, since "My Folder" has been closed, "Paste Up" (object 708), link 728 and link 727 are not shown. However, these still exist, but are not currently visible to a user.

In Figure 35, placing cursor 781 on the icon "Sample Text" and double clicking a button on mouse 20 results in "Sample Text" being opened. In Figure 36, an open window 789 for "Sample Text" is shown.

Figure 26A, Clipboard object 720 is shown to be a parent of "Paste Up" object 707 through a link 722.

In Figure 27, window 785 has been activated. From a menu 787, "Paste" is selected. The result, shown in Figure 28, is an icon 707a appearing in window 785, which indicates that "Paste Up" (object 707) is shared by window 785 and the NewWave Office window. In Figure 28A, as a result of the paste, "Paste Up" is now shown to be both a child of Clipboard 720 through link 722 and a child of "My Folder" 705 through a link 727. In Figure 29, showing just the interconnection of objects visible to the user, "Paste Up" (object 707) is shown to be a child of "My Folder" 705 through link 727. Since "Paste Up" (object 707) is shared, not copied, "Paste Up" (object 707) remains a child of NewWave Office through link 717.

One key feature of data links is automated data transfer. When a child object is open and the user changes a part of it which is "shared out", then it makes a call to OMF 100. OMF 100 checks if any of the object's parents "care" about this particular change. If they care and if they are also open, OMF 100 sends to the parents a message informing them that new data is available. The parent can then send messages to the child to produce or display the data. This feature allows the user to establish compound objects with complex data dependencies, and then have changes made to any sub-part be automatically reflected in other parts. For example, changing a number in a

The result is the opening and display of "Star" (object 706) in a window 796. Figure 50 shows the use of cursor 781 to select selection "Ellipse" in a menu window 797 which results in the data within "Star" (object 706) being changed from a star to an ellipse. As shown in Figure 51, the result is a change both in data displayed in window 796 and data displayed in region 795 of window 791.

In Figure 52, cursor 781 is used to define a region 797 in window 791. In Figure 53, cursor 781 is used to select a selection "Create a New..." in pull down menu 798. As a result of this selection dialog box 799 appears in Figure 54. Dialog box 799 contains icons for the two classes of objects available which are able to display data in region 797 of window 791. Using cursor 781, the icon "HP Shape" has been highlighted. Using keyboard 19 the name "New Shape" has been typed in as the name for a new object to be created. Cursor 781 now points to a regions labelled "OK". Once this region is selected, a new object titled "New Shape" is created. Data for "New Shape" is displayed in region 797 of window 791 as is shown in Figure 55. In Figure 56, "New Shape", (object 750) is shown to be a child of "Paste Up" (object 707) through a link 760.

In Figure 57 a window 800 for "New Shape" was opened by placing cursor 781 over region 797 of window 791 and clicking twice on a button on mouse 20. In Figure 58, cursor 781 is used to select the selection "Triangle" from a pull down menu 801. The result, as shown in Figure 59, is

In Figure 37 a window 791 for "Paste Up" (object 707) has been opened by double clicking on the icon for "Paste Up". In Figure 38, using Cursor 781, controlled by mouse 20, a portion 790 of the text of "Sample Text" has been selected. The portion in inverse video stating "New Wave Office environment" is portion 790.

In Figure 39, cursor 781 is used to select the selection "Share" in a pull down menu 792. In Figure 40, an area 793 in window 791 is selected using cursor 781. In Figure 41, a selection "Paste" is selected from a pull down menu 794 using cursor 781. In Figure 42, "Sample Text" is linked to "Paste Up" (object 707) and displayed text 790 is displayed in "Paste Up" window 791. In Figure 43 "Sample Text" (object 709) is shown to be a child of "Paste Up" (object 707) through a link 729. In Figure 42, displayed text 790 is shown in gray because "Star" window 789 is open. In Figure 44, "Star" window 789 is closed so displayed text 790 is clearly displayed.

In Figure 45, a region 795 of window 791 is selected using cursor 781. Figure 46 shows cursor 781 dragging the icon "Star" into region 795 of window 791.

In Figure 47, data from "Star" (object 706) is now displayed in region 795 of window 791. As may be seen in Figure 48, "Star" (object 706) is now a child of "Paste Up" (object 707) through a link 726.

In Figure 49, a user has placed cursor 781 over region 795 of window 791 and double clicked a button on mouse 20.

include the data identification number given to OMF 100 by child 709.

In Figure 40, area 793 in window 791 is selected using cursor 781. In Figure 41, a selection "Paste" is selected from a pull down menu 794 using cursor 781. At this point parent object 707 ("Paste Up") requests OMF 100 for a link making him the parent of what is on clipboard 720. The view specification record for the link between clipboard 720 and child 709 is copied for link 729 between parent 707 and child 709. In Figure 43 "Sample Text" (object 709) is shown to be a child of "Paste Up" (object 707) through link 729.

In Figure 42, "displayed text 790 is displayed in "Paste Up" window 791. In accomplishing this, parent object 707 makes a call to OMF 100 asking that a message be sent to the object identified by the reference name for link 729. This message requests the child object 709 to display data from this link into a location specified by parent object 707. OMF 100 takes the message from parent 707, adds the data identification number from the view specification record for link 729, and delivers the message to child 709. Child 709 displays the data in the specified location, in this case area 793. The name of the message sent from parent 707 to OMF 100 to child 709 is "DISPLAY_VIEW", further described in Appendix B, attached hereto.

Another message "PRINT_SLAVE", also described in Appendix B, may be used when it is desired to print data on a printer rather than display data on a terminal screen.

In addition, Parent 707 may send a "GET_SIZE" message to child object 709. In a "GET_SIZE" message, parent object 707 identifies a reference name for link 729 and indicates coordinates for a display. OMF 100 takes the GET_SIZE message from parent 707, adds the data identification number from the view specification record for link 729, and delivers the message to child 709. Child 709 returns to parent 707 the size of the portion of the specified area that child 709 would use to display the data. This allows parent 707 to modify the region reserved for displaying data from child 709 when child 709 is not able to scale the data to fit in the region specified by parent 707.

When data from a child object is being displayed by a parent object, and the child object changes the displayed data, the child objects notifies OMF 100 that there has been a change in the data object. For example, as described above, in Figure 47, data from "Star" (object 706) now displayed in region 795 of window 791. And, as may be seen in Figure 48, "Star" (object 706) is a child of "Paste Up" (object 707) through a link 726. Since data is being passed from child object 706 to parent object 707, link 726 is a data link which includes a view specification.

In Figure 49, the method for changing data in child object 706 is shown. A user places cursor 781 over region 795 of window 791 and double clicks a button on mouse 20. The result is the opening and display of "Star" (object 706) in a window 796. Using cursor 781 to select selection

"7". Since object 671 is a folder, there is no view, therefore the three bits within subcolumn 734 would be "000".

In Figure 78, Object 676 is a document and has a tag of "17". Object 676 is a parent of an object 677 through a link 679 and a parent of an object 678 through a link 680. Object 677 has a tag of "8". Link 679 as a reference name "1". Object 678 has a tag of "21". Link 680 has a reference name "3".

In Figure 79, an entry 737 describes link 679 shown in Figure 78. That is, in column 731 of entry 737 there is the parent tag "17". In column 732 there is the child tag "8" and in column 733 there is the reference name "1". Object 676 is a document, and assuming there is a view associated with link 679, the three bits within subcolumn 734 contain the three bits "110" and entry 738 is a view specification record.

Similarly, an entry 739 describes link 680 shown in Figure 78. That is, in column 731 of entry 739 there is the parent tag "17". In column 732 there is the child tag "21" and in column 733 there is the reference name "3". Assuming there is a view associated with link 680,.the three bits within subcolumn 734 contain the three bits "110" and entry 740 is a view specification record.

In Figure 80, view specification record 740 is shown to include a field 741 which contains a data identification for the view, a field 742 which indicates whether there is a

snapshot used in the view, and a field 743 which contains miscellaneous information about the view. The data identification number is used by the child object of the link, to determine what data is sent through the link.

Figures 37 - 43 show the establishment of a link with a view. As has been discussed before, in Figure 37 window 791 for "Paste Up" (object 707) has been opened by double clicking on the icon for "Paste Up". In Figure 38, using Cursor 781, controlled by mouse 20, portion 790 of the text of "Sample Text" has been selected. The portion in inverse video stating "New Wave Office environment" is portion 790.

In Figure 39, cursor 781 is used to select the selection "Share" in a pull down menu 792. Once "Share" is selected, child object 709 ("Sample Text") creates a data identification number which identifies portion 790 of the text to child object 709. Child object 709 also causes OMF 100 to put a link to child object 709 on clipboard 720--Child object 709 communicates to OMF 100 through command set forth in Appendix B, attached hereto--. Child object 709 also informs OMF 100 what data identification number is associated with the new link between the child 709 and clipboard 720. If there is a snapshot associated with the link, child 709 will also inform OMF 100 if there is a snapshot associated with the link. Snapshots are discussed more fully below. As a result OMF 100 will make an entry in HPOMF.XRF 603 for a link between clipboard 720 and child object 709. The view specification record for the link will

VS_TEXTDISKLOC	File position in HPOMF.PRP where a view's 32 character textual data ID is located. This contains zero if no textual data ID has been defined by the child. The low order five bits of the file position are always zero and are thus not stored in the Misc field. The hexadecimal number OFFF FFE0 is a mask which indicates which bits are used for this bit field.
VS_INITIALIZED	Set if the view specification has been initialized. If clear, all information in the view specification is zero. The hexadecimal number 0000 0010 is a mask which indicates which bits are used for this bit field.
VS_RESERVED	Reserved for future expansion. The hexadecimal number 0000 0008 is a mask which indicates which bits are used for this bit field.
VS_VIEWCLASS	Specifies the view class the child assigned to the view. The view class defines what view methods are available to the parent. The hexadecimal number 0000 0007 is a mask which indicates which bits are used for this bit field.

For example, in Figure 77, Object 671 is a folder and has a tag of "6". Object 671 is a parent of an object 672 through a link 674 and a parent of an object 673 through a link 675. Object 672 has a tag of "12". Link 674 as a reference name "1". Object 673 has a tag of "19". Link

675 has a reference name "7". Reference names are picked by the parent object and need to be unique for the particular parent object; however, other parents may have a link with the same reference name provided each reference name is unique for each parent.

Figure 79 shows a block diagram of HPOMF.XRF 603. HPOMF.XRF contains an entry for each link between parents and children. In HPOMF.XRF 603 column 731 contains the tag of the parent for each link. Column 732 contains the tag of the child for each link. Column 733 contains the reference name for each link. The first three bit positions of column 733, shown in Figure 79 as sub-column 734, indicate whether a view specification file record is present ("110") whether no view specification file record follows ("000") or whether the link is between is a link from the global parent to a global object ("100").

As may be seen, entry 735 is an entry which describes link 674 shown in Figure 77. That is, in column 731 of entry 735 there is the parent tag "6". In column 732 there is the child tag "12" and in column 733 there is the reference name "1". Since object 671 is a folder, there is no view, therefore the three bits within subcolumn 734 would be "000".

Similarly, entry 736 is an entry which describes link 675 shown in Figure 77. That is, in column 731 of entry 736 there is the parent tag "6". In column 732 there is the child tag "19" and in column 733 there is the reference name

records cannot be identified by the content within a view specification file record. HPOMF.XRF is increased in size 16K bytes at a time. A newly allocated portion of HPOMF.XRF is filled with zeros. File records within HPOMF.XRF which are free or which define a link have the following fields listed in Table 5:

Table 5

ParentTag	Contains the tag (HPOMF.CAT record number) of the parent object of this link. If this field is 0, then this record does not define a link and is free.
ChildTag	Contains the tag of the child object of this link. If ParentTag in this record is 0, and this field is also 0, then no record beyond this record in HPOMF.XRF defines a link.
RefName	Contains the reference name that the parent has assigned to the link. This field is meaningless if ParentTag or ChildTag is zero. Otherwise, if the top three bits of this value are 110, the next record in the file is a view specification.

File records within HPOMF.XRF which are view specification file records have the following fields listed in Table 5A:

Table 5A

DataId	Contains the value that the child has assigned to identify the part of itself that is being viewed through the link.
Snapshot	Contains the tag (HPOMF.CAT record number) of the object which is the view's snapshot, or if zero, the view has no snapshot. For further discussion of snapshots, see below.
Misc	Composed of several bit fields described below:
VS_NEWDATASET	Set if child has told OMF that new data is available, but has not been announced to the parent. The hexadecimal number 8000 0000 is a mask which indicates which bits are used for this bit field.
VS_NEWDATAANNOUNCED	Set if child has told OMF to announce new data to parent, but parent was inactive and was not notified. The hexadecimal number 4000 0000 is a mask which indicates which bits are used for this bit field.
VS_SNAPSHOTOLD	Set if child has told OMF that the view's snapshot is out-of-date. The hexadecimal number 2000 0000 is a mask which indicates which bits are used for this bit field.
VS_WANTMESSAGES	Set if child has told OMF that it wants to process view messages when snapshot is out-of-date. The hexadecimal number 1000 0000 is a mask which indicates which bits are used for this bit field.

Table 4

lFirstFreeEntry	Is "-1" if this record defines an installed class, otherwise this record is free and this field is the record number of the next free record, or "0" if there are no more free records. If the record is free, none of the other fields in the record is meaningful.
ModuleFileName	Specifies the name of the application associated with objects of this class as a null-terminated string.
properties	Specifies the number of properties, the length of the property names and the location in HPOMF.PRP of the object's properties. See the description of HPOMF.PRP below for further definition of the structure of this field.

In Figure 75, the relationship of HPOMF.CAT and HPOMF.CLS is shown. Within each object entry within HPOMF.CAT, the record number, which is an object's tag, serves as an identifier 650 of data files in a mass storage memory 170 associated with the object. The field "TypeInClass" serves as an identifier 651 of the class entry in HPOMF.CLS, which identifies the class of each object. Within each class entry in HPOMF.CLS, the field "ModuleFileName" serves as an identifier 652 of the application file in mass storage memory 170 which is associated with the class.

In Figure 76, the organization of a portion of mass storage memory 170 is shown. A root directory 660 contains pointers to an HPNWDATA directory 661 and HPNWPROC directory

668. HPNWPROG directory 668 is the location of storage for applications files, represented by arrows 669. HPNWDATA contains a plurality of HPOMFddd directories, represented by directories 662, 663, 664, 665 and 666. In the HPOMFddd directories are stored data files associated with objects. The "ddd" in HPOMFddd stands for a three digit, leading zeros, hexadecimal number. Each HPOMFddd directory has a different "ddd" hexadecimal number. The "ddd" number indicates which HPOMFddd directory stores data files for a particular object. Data files for a particular object are stored in the HPOMFddd directory which has a "ddd" number equal to the tag for the object divided by an integer number, e.g., fifty four. Within each HPOMFddd directory, files are stored by tag numbers, e.g. data file names have the format xxxxxxxx.lll, where "xxxxxxxx" is an eight digit leading zeros hexadecimal tag, and "lll" are a reference chosen by the application.

System file 603, HPOMF.XRF is also referred to as SYSXREF. This file is a list of all the links existing in the system. It is record oriented, but does not have a header record. Each record file is either free, or defines an existing link, or is used as an overflow record from the previous record to specify additional view specification information. Records that contain view specifications are called view specification file records. View specification file records can be identified only by a previous record which defines an existing data link; view specification file

Table 2

lFirstFreeEntry	Is "-1" if this record defines an object, otherwise this record is free and this field is the record number of the next free record, or "0" if there are no more free records. If the record is free, none of the other fields in the record is meaningful.
TypeInClass	Specifies the class of this object. This is the number of the record in HPOMF.CLS that indicates to which class the object belongs (see discussion of class above).
SysCatFlags	Specifies if the object is global if the bit masked by the number 20 (hexadecimal) is set in this byte. In the preferred embodiment all other bit positions must contain "0" and are not used.
properties	Specifies the number of properties, the length of the property names and the location in HPOMF.PRP of the object's properties. See the description of HPOMF.PRP below for further definition of the structure of this field.
fastprops	Certain object properties, such as name, are so heavily accessed that they are stored directly in this field, rather than indirectly in the properties file. Properties stored in this field are called "fast properties."

System file 602, HPOMF.CLS is also referred to as SYSCLASS. This system file is a list of all installed classes in the system. It is record oriented. The first record, numbered 0, is a header which contains various

signatures (see above) and is used to manage a list of free records. All other records either define an installed class or are free. In the preferred embodiment HPOMF.CLS can grow dynamically, but cannot shrink.

Each file record in HPOMF.CLS is thirty-two bytes in length. HPOMF.CLS file record 0 (the header) contains the following fields listed in Table 3:

Table 3

lFirstFreeEntry	Contains the record number of the first free record in HPOMF.CLS, or "0" if there are no free records.
FileId	Contains the null terminated string "HPOMF.CLS"
Version	Contains the file format version number.
lMaxRecordNumber	Contains the number of the highest record ever allocated from within HPOMF.CLS (this highest record may or may not be free).

Table 4, below, contains the fields for file records in HPOMF.CLS for file records other than file record 0:

other programs running on computer 18. System files 601-607 serve as a data base that provides various information. They provide information about object properties such as what class each object is what is the name of each object. System files 601-607 provide information about classes of objects such as what application is associated with each class of objects, what icon represents objects of a particular class and lists of what messages (such as those shown in Figure 2) can be processed by objects of a particular class. System files 601-607 also contain information about links between parent and child objects including a list of parents and reference names of each link from a parent for each object; a list of children and reference names of each link to a child for each object; and additional information to manage data exchange across data links. Additionally, system files 601-607 contain general information such as what files are installed in the operating system for each class that is installed, and what objects have requested automatic restart when the OMF 100 is restarted.

In the preferred embodiment of the present invention system file 601 is referred to as HPOMF.CAT, system file 602 is referred to as HPOMF.CLS, system file 603 is referred to as HPOMF.XRF, system file 604 is referred to as HPOMF.PRP, system file 605 is referred to as HPOMF.INS, system file 606 is referred to as HPOMF.SDF and system file 607 is referred

that a triangle is now displayed both in window 800 and in region 797 of window 791.

In Figure 60, window 800 has been closed. In Figure 61, "New Shape" is selected by placing cursor 781 over region 797 of window 796, and clicking a button on mouse 20. In Figure 62, cursor 781 is used to select selection "Share" from pull down menu 794. In Figure 63, cursor 781 is used to select a region 802 of window 791. In Figure 64, cursor 781 is used to select selection "Paste" from pull down menu 794. The result, as shown in Figure 65, is the sharing of "New Shape" with data from "New Shape" being displayed in region 797 and in region 802 of window 791. In Figure 66, "New Shape" (object 750) is shown to have an additional link 770, from its parent "Paste Up" (object 707).

In Figure 67, region 797 has been selected using cursor 781. Cursor 781 is then used to select selection "Cut" from pull down menu 794. The result, as seen in Figure 68, is that region 781 has been removed from window 791. In Figure 69, cursor 781 is used to select selection "Paste" from pull down menu 783. The result, shown in Figure 70, is an icon for "New Shape", pointed to by cursor 781. In Figure 71, "New Shape (object 750) is shown to now be a child of NewWave Office (object 100), through a link 780.

In Figure 72, OMF 100 is shown to contain seven system files: system file 601, system file 602, system file 603, system file 604, system file 605, system file 606 and system file 607. OMF interface 599 serves as interface of OMF to

global object, is a descendent of global object. Although Figure 74 shows only six global objects, the number of global objects operating on a system is a matter of system configuration. Any object in the system can refer to a global object by ~~by~~ using the reference name of the link to that global object from the global parent.

As may be seen from Figure 73, file records in HPOMF.CAT are numbered consecutively. These numbers serve as tags, which identify each object.

In the preferred embodiment of the present invention, each record is 128 bytes in length. The fields for file record 0 are listed in Table 1 below:

Table 1

lFirstFreeEntry	Contains the record number of the first free record in HPOMF.CAT, or "0" if there are no free records.
FileId	Contains the null terminated string "HPOMF.CAT". This serves as a signature.
Version	Contains the file format version number, which also serves as a signature.
lMaxRecordNumber	Contains the number of the highest record ever allocated from within HPOMF.CAT (this highest record may or may not be free).

Table 2, below, contains the fields for file records in HPOMF.CAT for file records other than file record 0:

to as HPOMFICO.NWE. A description of each system file is now given.

System file 601, HPOMF.CAT, is also referred to as SYSCAT. HPOMF.CAT is a catalog of all the existing objects in the system. In Figure 73, HPOMF.CAT is shown to be record oriented. HPOMF.CAT has a plurality of file records. In Figure 73, file record 0 through file record 8 are shown, although HPOMF.CAT may contain many more file records than are shown in Figure 73. File record 0 is a header which contains various signatures and is used to manage a list of free file records. A signature is some known value which if present indicates that the file is not corrupted. File record 1 through file record 8 and additional file records (not shown) either define an existing object, or are free. In the preferred embodiment HPOMF.CAT can grow dynamically, as more file records are needed, but cannot shrink.

File record 1 defines a special object called the global parent. The global parent has a form different than every other object, and may be regarded as a "pseudo" object. Figure 74 shows the global parent to be the parent of global object 250 through link 260, global object 251 through link 261, global object 252 through link 262, global object 253 through link 263, global object 254 through link 264 and global object 255 through link 265, as shown. Global objects 250-255 are also within HPOMF.CAT. Each global object 250-255 may be a parent of one or more objects in HPOMF.CAT. Each object in HPOMF.CAT which is not a

"Ellipse" in a menu window 797 results in the data within "Star" (object 706) being changed from a star to an ellipse. As shown in Figure 51, the result is a change both in data displayed in window 796 and data displayed in region 795 of window 791.

Child object 706 accomplishes this change by making a call to OMF 100 stating that data associated with the data identification number associated with link 726 is changed. OMF 100 looks up all of the links that use the data identification number. If the parent object of any of the links is not active, OMF 100 sets the bit VS_NEWDATAANNOUNCED for that link in HPOMF.XRF. When the parent object is activated, the parent object can then request the new data.

If the parent object is active, OMF 100 will send a message to the parent object saying that new data is available. OMF 100 will identify to the parent object the reference name of the link for which there is additional data. The parent object sends a message to the child object if it wants the new data displayed. In the present case parent object 707 is active , and has requested the new data to be displayed in region 795 of window 791. A further description of the View Specifications are found in Appendixes B, C and D.

The advantage of the present invention is that parent object 707 is able to communicate with child object 706 through OMF 100, without parent object 707 or child object

706 knowing the identity or any other details about each other. The parent object identifies the link using only the reference name of the link. The child object identifies the link using just the data identification number of the link. OMF 100 does all the translation and identification of which links and which objects are involved.

System file 604, HPOMF.PRP, is also referred to as SYSPROP. HPOMF.PRP contains all the object and class properties except for the fast object properties which are contained in HPOMF.CAT. Each record in system file 601 (HPOMF.CAT) and system file 602 (HPOMF.CLS) has a properties field, as described above. Each properties field contains the fields described in Table 6 below:

Table 6

DirDiskLoc	Contains the position (byte offset) within HPOMF.PRP of the property list directory.
nProps	Contains the number of properties in the property list. This is the number of entries in the directory entry array described below.
PoolSize	Contains the combined length of all the names of the properties in the property list, including a null-terminating byte for each name. This is the size of the directory name pool described below.

For each object and for each class, at the DirDiskLoc position in the HPOMF.PRP file is the property directory for that object or that class. The directory has two major

portions: the entry array, followed by the name pool. The entry array has one entry for each property in the property list. Each entry has fields set out in Table 7 below:

Table 7

ValueLen	Specifies the length in bytes of the associated property. This can be zero.
ValueDiskLoc	Contains the position within HPOMF.PRP of the value of the associated property. If ValueLen is zero, this is also zero, and there is no value stored anywhere.
CacheOffset	This field is only used at run time and is not meaningful in the file.

Immediately following the entry array is the name pool. This portion of HPOMF.PRP contains the null-terminated names of properties in the property list, in the same order as the entry array. Properties may include such things as titles, user comments, date and time of creation, the user who created the object, etc. For more information on properties, see Appendix D.

HPOMF.PRP grows dynamically as need. At the beginning of HPOMF.PRP there is a 128 byte bitmap which controls the allocation of the first 1024 pages of HPOMF.PRP. Each page is 32 bytes in length. These pages immediately follow the bit map. The bitmap is an array of words with the most significant bit of each word used first. Thus, bits 15 through 0 of the first word of the bitmap control the allocation of pages 0 through 15 of the file, respectively.

When storage in the first 1024 pages is insufficient, a second bitmap is added to the file following page 1023. This bitmap controls the allocation of pages 1024 through 2047, which immediately follow the second bitmap. Additional bitmaps and pages are added in the same way, as needed.

Each directory and property value is stored as a single block in the file, i.e., as a contiguous run of pages that are all allocated in the same bitmap. This causes the restriction that no directory or value can exceed 32K bytes (1024 times 32) in length.

System file 605, HPOMF.INS, is also referred to as SYSINSTL. HPOMF.INS contains a list of the files that were copied to the system when each class was installed. This information is used so that these files can be deleted when the class is de-installed.

The very beginning of HPOMF.INS is a double word value which serves as a validity/version identifier. In the preferred embodiment the value of this double word must be 0101ABCD hex to be valid. In Table 8, this number is stored as shown because of the protocols for storage in the particular processor used by the preferred embodiment, i.e. an 80286 microprocessor made by Intel Corporation.

Following the double word comes a series of variable length records. There is one record for each installed class. The first word of each record is the length of the rest of the record, in bytes. This is followed by the null-

terminated class name of the installed class. Then follows the file names of the files copied to the OMF directories, each terminated by a null byte, and preceded by a byte which gives the length of the file name, including the length byte and the null terminator. If the file name begins with the special character "*", the file is assumed to be located in the HPNWPROG directory. If the file name begins with the special character "+" the file is assumed to be located in the HPNWDATA directory.

For example, assume two classes are installed: class "AB" and class "CDE". Class "AB" caused two files to be installed: "Z" to HPNWPROG directory 668 and "YY" to the HPNWDATA directory. Class "CDE" caused 1 file to be installed: "XXX" to HPNWPROG directory 668. Given this case Table 8 below shows the contents of HPOMF.INS for this example:

Table 8

offset	content	comments
0	CD AB 01 01	File header/version check
4	0C 00	Length of AB record (12 decimal)
6	41 42 00	"AB" + Null
9	04	Length of length byte "*Z" + Null
A	2A 5A 00	"*Z" + Null
D	05	Length of length byte + "+YY" + Null
E	2B 59 59 00	"+YY" + Null
12	0A 00	Length of CDE record (10 decimal)
14	43 44 45 00	"CDE" + Null
18	06	Length of length byte + "*XXX" + Null
19	2A 58 58 58 00	"*XXX" + Null

System File 606, HPOMF.SDF is also referred to as the "shutdown file". HPOMF.SDF exists only when the system has been cleanly shut down. It is deleted as the system starts, and created as it shuts down. On startup, if this file is missing, OMF assumes that the last session ended abnormally, and so it goes through its crash recovery procedures to validate and repair the system files as best it can. The system files can be in an invalid but predictable state on a crash. These errors are corrected without user intervention. Certain other kinds of file consistency errors are detected, but are not really possible from an "ordinary" system crash. These errors are in general not correctable and the OMF will not allow the system to come up in this case.

If HPOMF.SDF is present, it contains a list of objects. When the system is being shut down normally, each object which is active at the time can request that the OMF restart them when the system is restarted. The list of objects, then is the list of tags of objects which have requested that they be restarted when the system is restarted.

The first word in HPOMF.SDF is a flag word. If this word is non-zero, OMF will execute its crash recovery code even though HPOMF.SDF exists. Normal shutdown will set this flag when producing the file if some serious error occurred in the session being ended.

After the first word, the rest of the file is a sequence of three byte records. The first two bytes of each

record contain the tag of the object to be restored. The least significant byte is first. The third byte is not used in the preferred embodiment, and is zero.

For example, if the system is shut down cleanly in the last session and two objects, having tags of 2 and 7, respectively, have requested restart, the contents of HPOMF.SDF will be as set out in Table 9 below.

Table 9

offset	content	comments
0	00 00	Indicates no crash recovery needed
2	02 00	Tag of first object to restart
4	00	Unused and reserved
5	07 00	Tag of second object to restart
7	00	Unused and reserved

System file 7, HPOMFICO.NWE, is a Microsoft Windows dynamic library executable file which contains a dummy entry point and no data. Microsoft Windows is a program sold by Microsoft Corporation, having a business address at 16011 NE 36th Way, Redmond, WA 98073-9717. HPOMFICO.NWE also contains as "resources" the icons of each installed class. OMF modifies HPOMFICO.NWE directly during run time, and loads and unloads it to get the icon resources from it. The format of HPOMFICO.NWE is defined in Microsoft Windows documentation distributed by Microsoft Corporation.

Normally working with a view (see discussion on views above) causes a child's application to be invoked. Where large applications are involved, this can cause a lot of

unnecessary overhead. The use of snapshots allow this overhead to be eliminated.

A snapshot is an object that uses executable code from a separate library referred to as a dynamic access library (or DAL) rather than using the full application executable code. The only data file associated with a snapshot contains data which is to be sent from a child object to a parent object. The code which encapsulates the data file although referred to as a dynamic library, is still stored in directory HPOMFPROG (directory 668).

For example, Figure 81 shows a parent object 501 linked to a child object 502 through a link 504. Associated with link 504 is a snapshot 503. Once child object has designated snapshot 503 in a view specification record for link 504, snapshot 503 is able to provide data from child object 502 to parent 501 without the necessity of invoking an application associated with child object 502.

As shown in Figure 82, when there is no snapshot, child object 502 must be active in order to send view data 522 to parent object 501, in order for parent object 501 to display view data 522 in a window display 521. In Figure 83, however, snapshot 503 is shown to provide view data 522 to parent object 501 without the necessity of child 502 being active. Further implementation details of snapshots are given is Appendix B, Appendix C and Appendix D.

Appendix A is a list of major data structures within OMF 100.

Appendix B is a description of functions which OMF interface 599 recognizes in the preferred embodiment of the present invention.

Appendix C (HP NewWave Environment: Program Design Examples) Gives examples of how the preferred embodiment of the present invention may be implemented.including detail as to how OMF 100 allows data to be viewed between windows displayed on monitor 14.

Appendix D (Chapter 2 of Programmer's Guide) gives a further overview of the preferred embodiment of the present invention. further detail as to the operation of the preferred embodiment of the present invention.

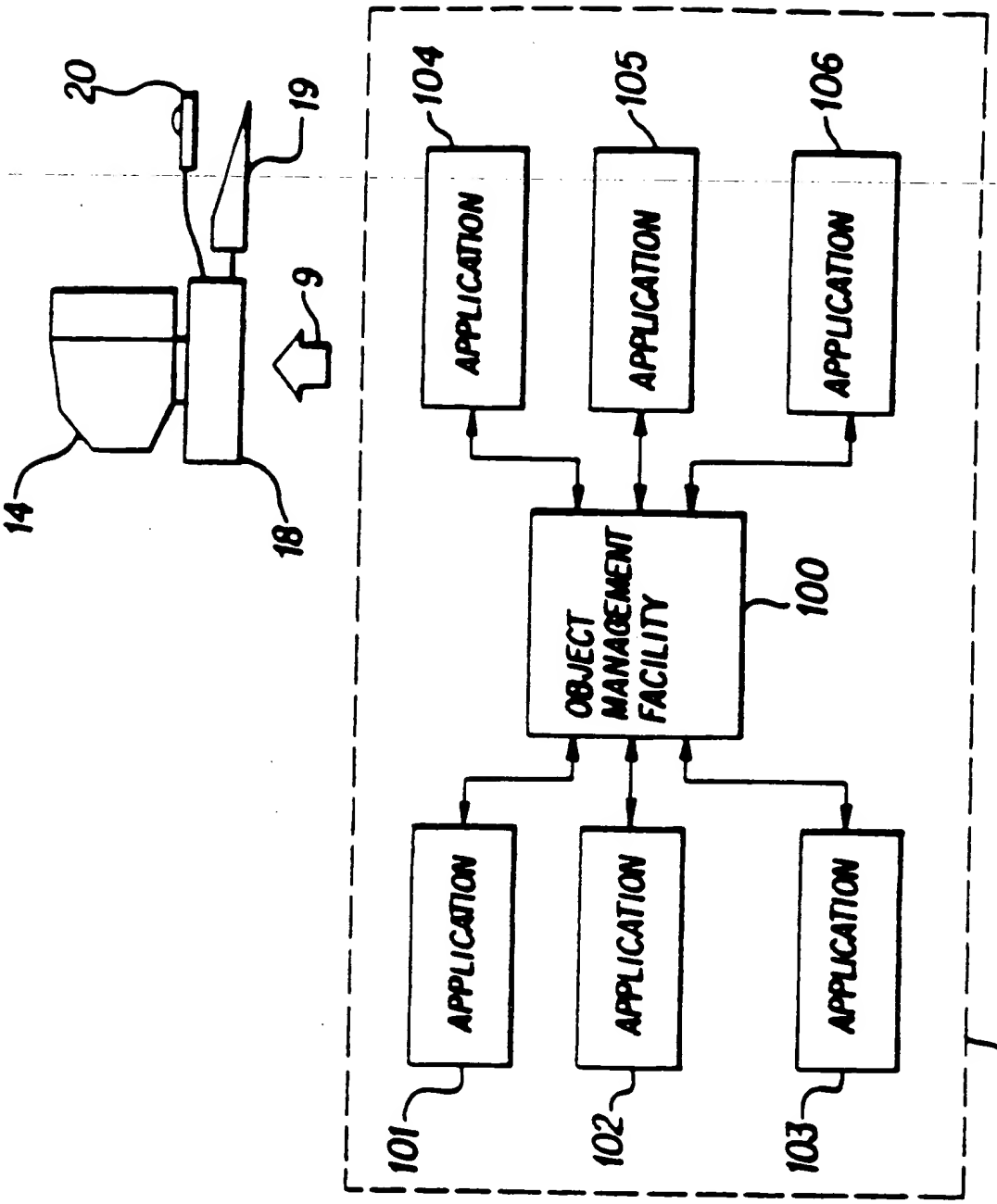

20
19
9
14
18
104
105
106
APPLICATION
APPLICATION
APPLICATION
OBJECT MANAGEMENT FACILITY
100
APPLICATION
APPLICATION
APPLICATION
101
102
103
17

FIG. 1

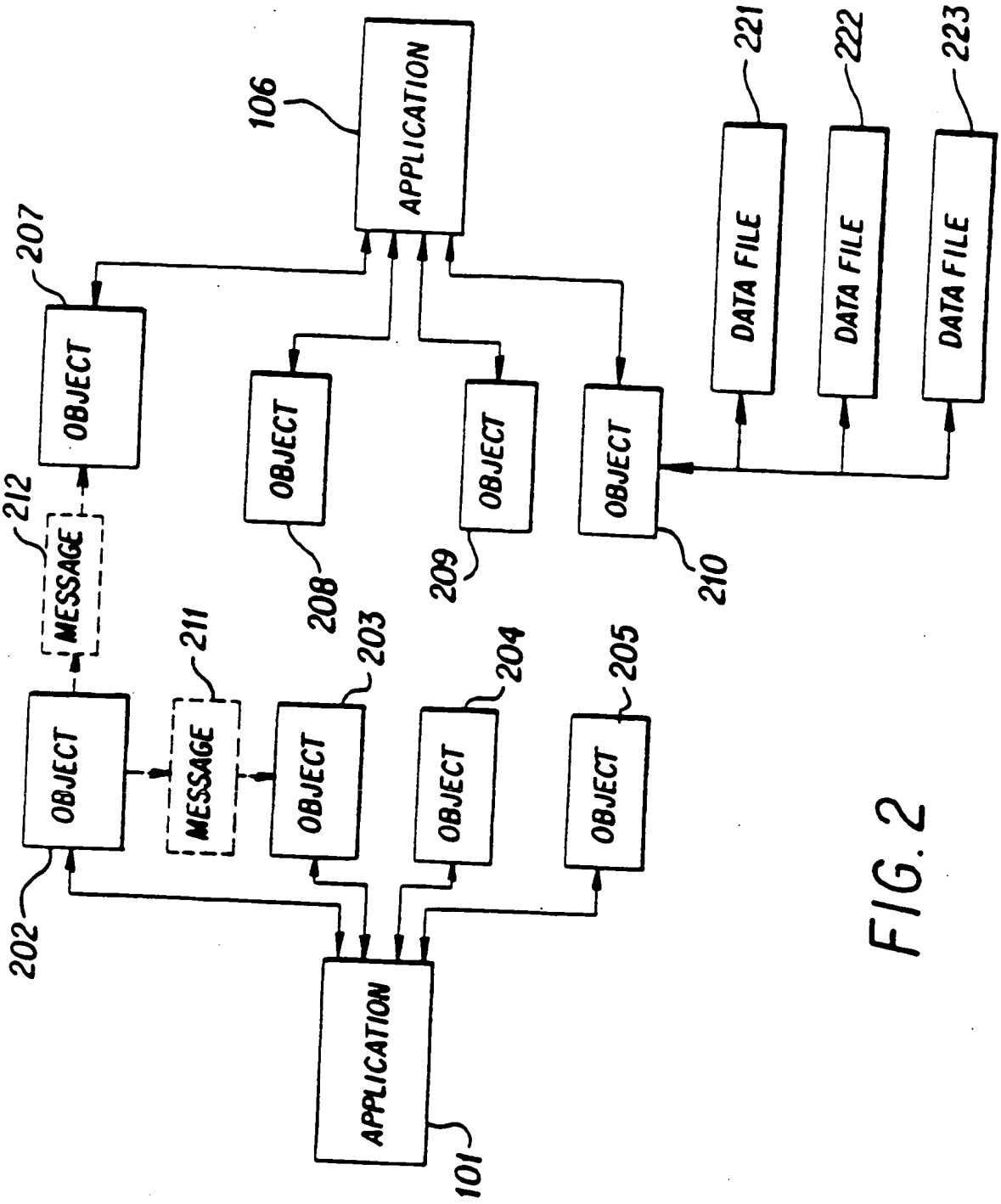

APPLICATION
106
DATA FILE
221
DATA FILE
222
DATA FILE
223
OBJECT
207
OBJECT
208
OBJECT
209
OBJECT
210
MESSAGE
212
MESSAGE
211
OBJECT
202
OBJECT
203
OBJECT
204
OBJECT
205
APPLICATION
101

FIG. 2

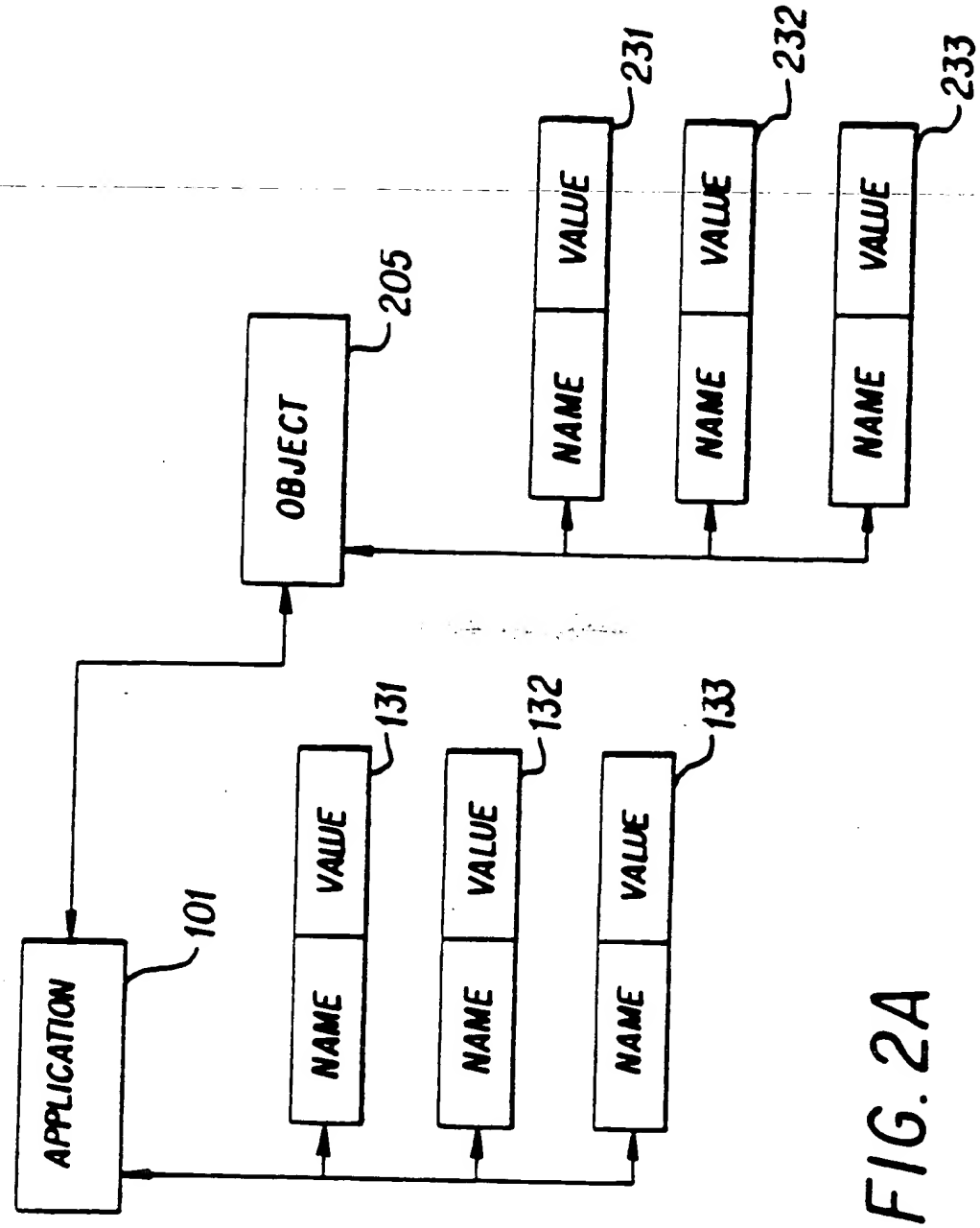

APPLICATION
101
OBJECT
205
NAME
VALUE
131
132
133
231
232
233

FIG. 2A

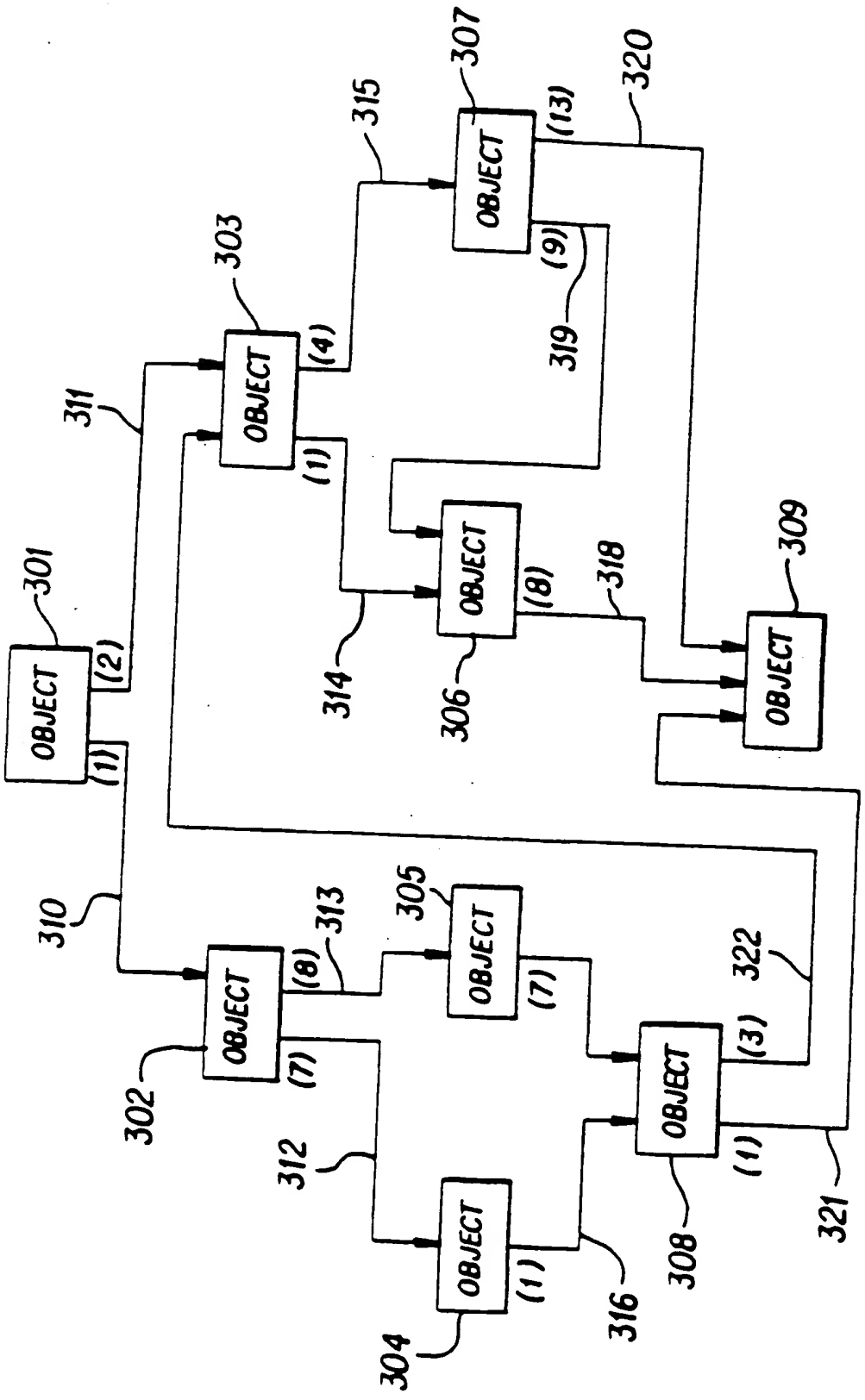

OBJECT
(1)
(2)
(4)
(7)
(8)
(9)
(13)
(3)
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
318
319
320
321
322

FIG. 3

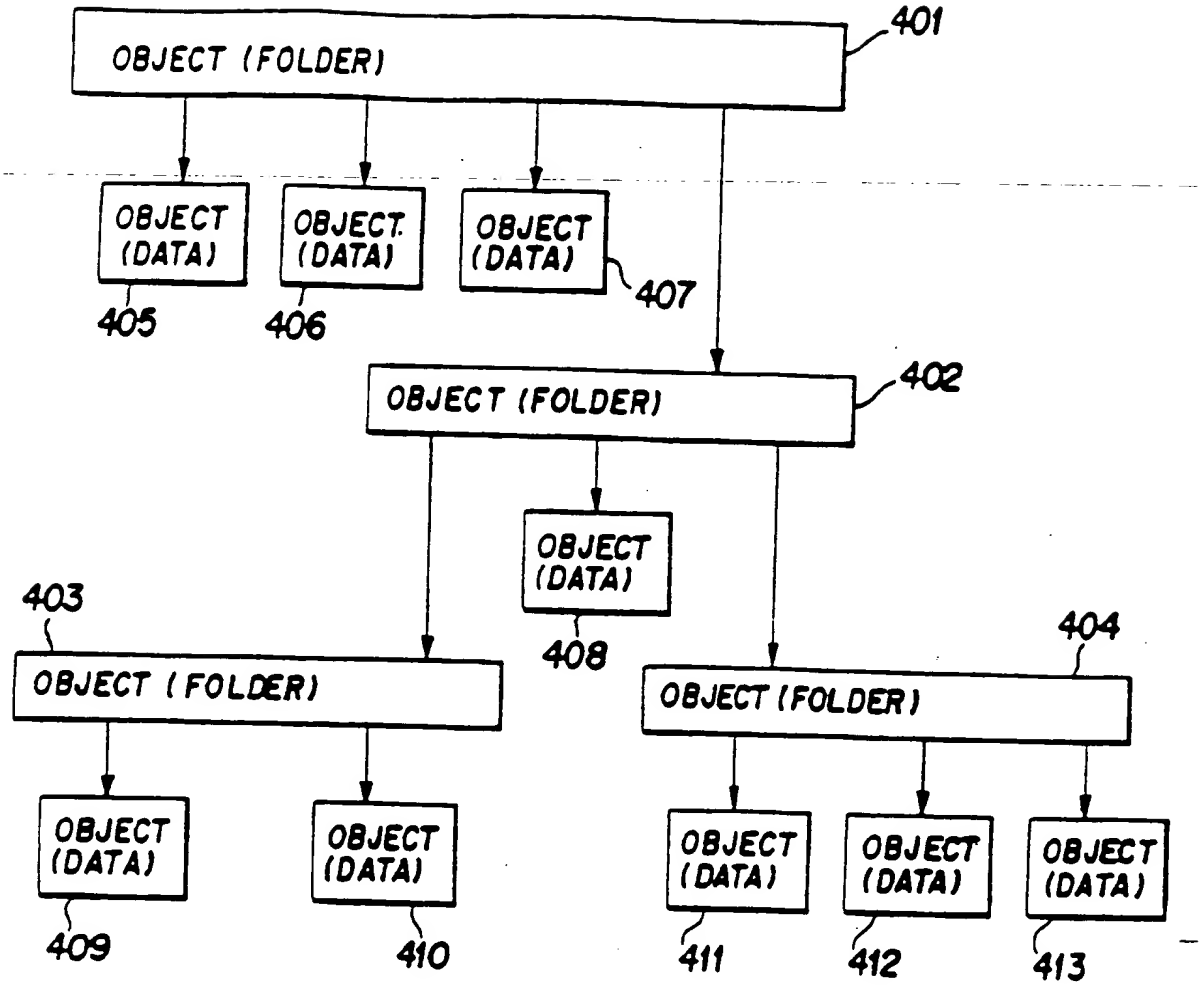
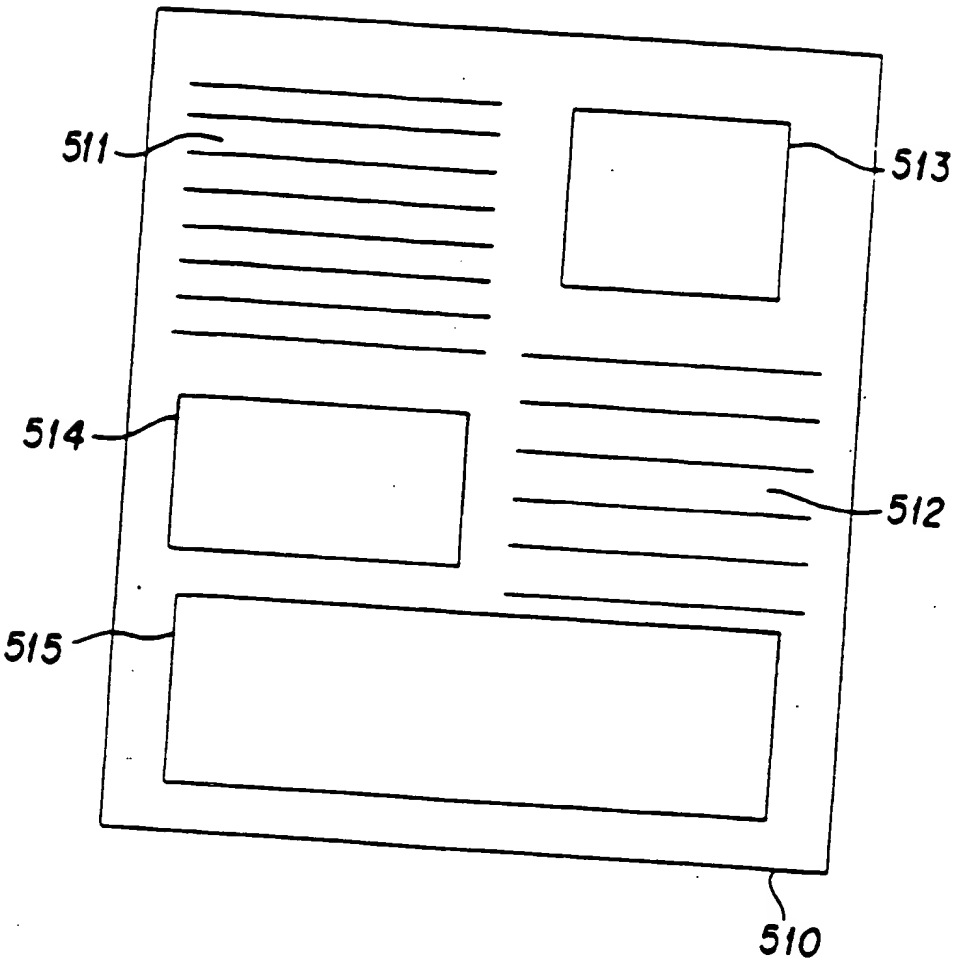

OBJECT (FOLDER)
401
OBJECT (DATA)
405
OBJECT (DATA)
406
OBJECT (DATA)
407
OBJECT (FOLDER)
402
OBJECT (DATA)
408
403
OBJECT (FOLDER)
404
OBJECT (FOLDER)
OBJECT (DATA)
409
OBJECT (DATA)
410
OBJECT (DATA)
411
OBJECT (DATA)
412
OBJECT (DATA)
413

FIG. 4

FIG. 5



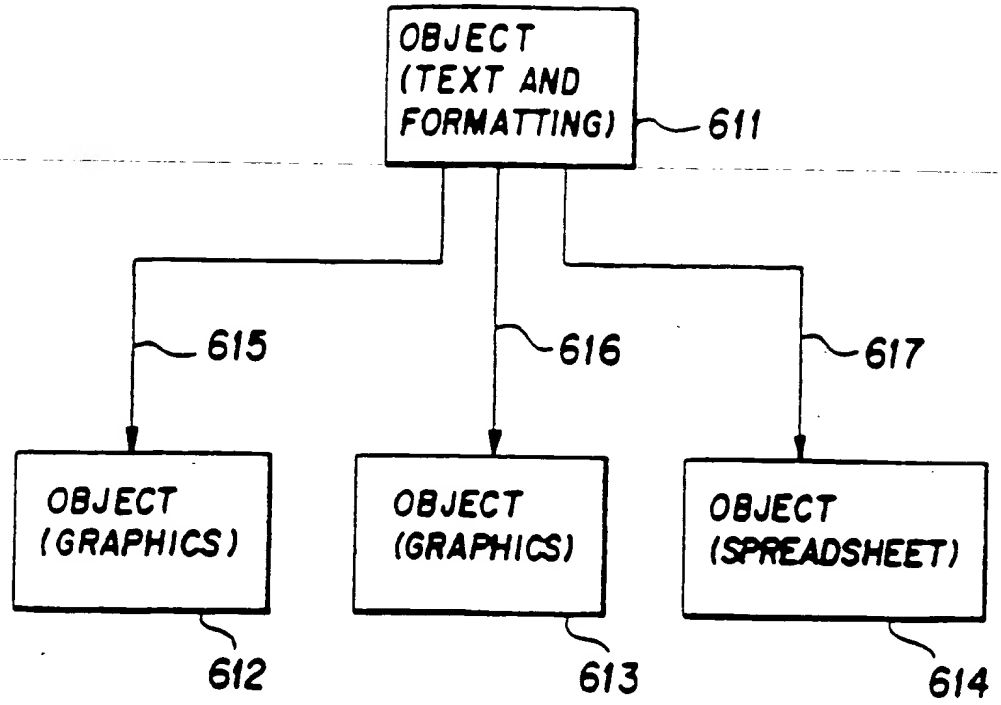

OBJECT
(TEXT AND
FORMATTING)
611
615
616
617
OBJECT
(GRAPHICS)
OBJECT
(GRAPHICS)
OBJECT
(SPREADSHEET)
612
613
614

FIG. 6

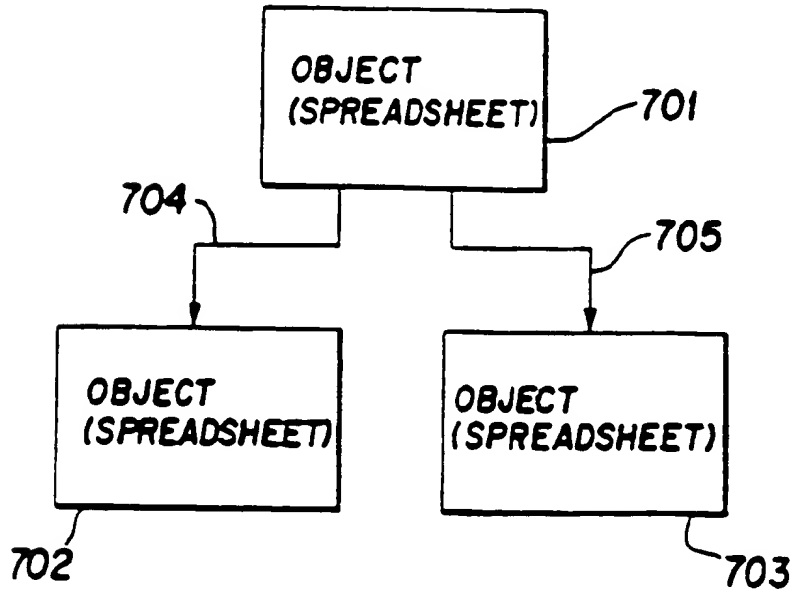

OBJECT
(SPREADSHEET)
701
704
705
OBJECT
(SPREADSHEET)
OBJECT
(SPREADSHEET)
702
703

FIG. 7

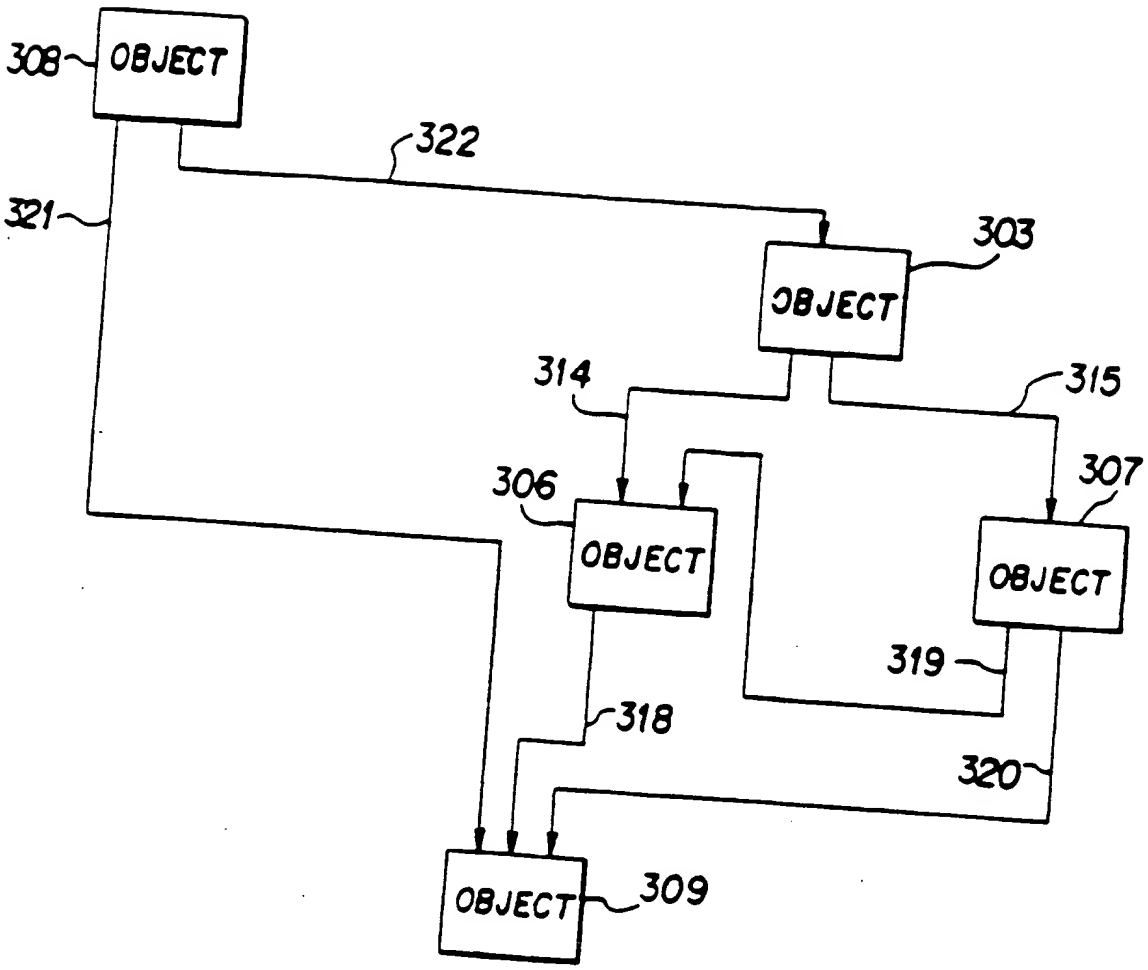

308
OBJECT
322
321
303
OBJECT
314
315
306
OBJECT
307
OBJECT
319
318
320
OBJECT
309

FIG. 8

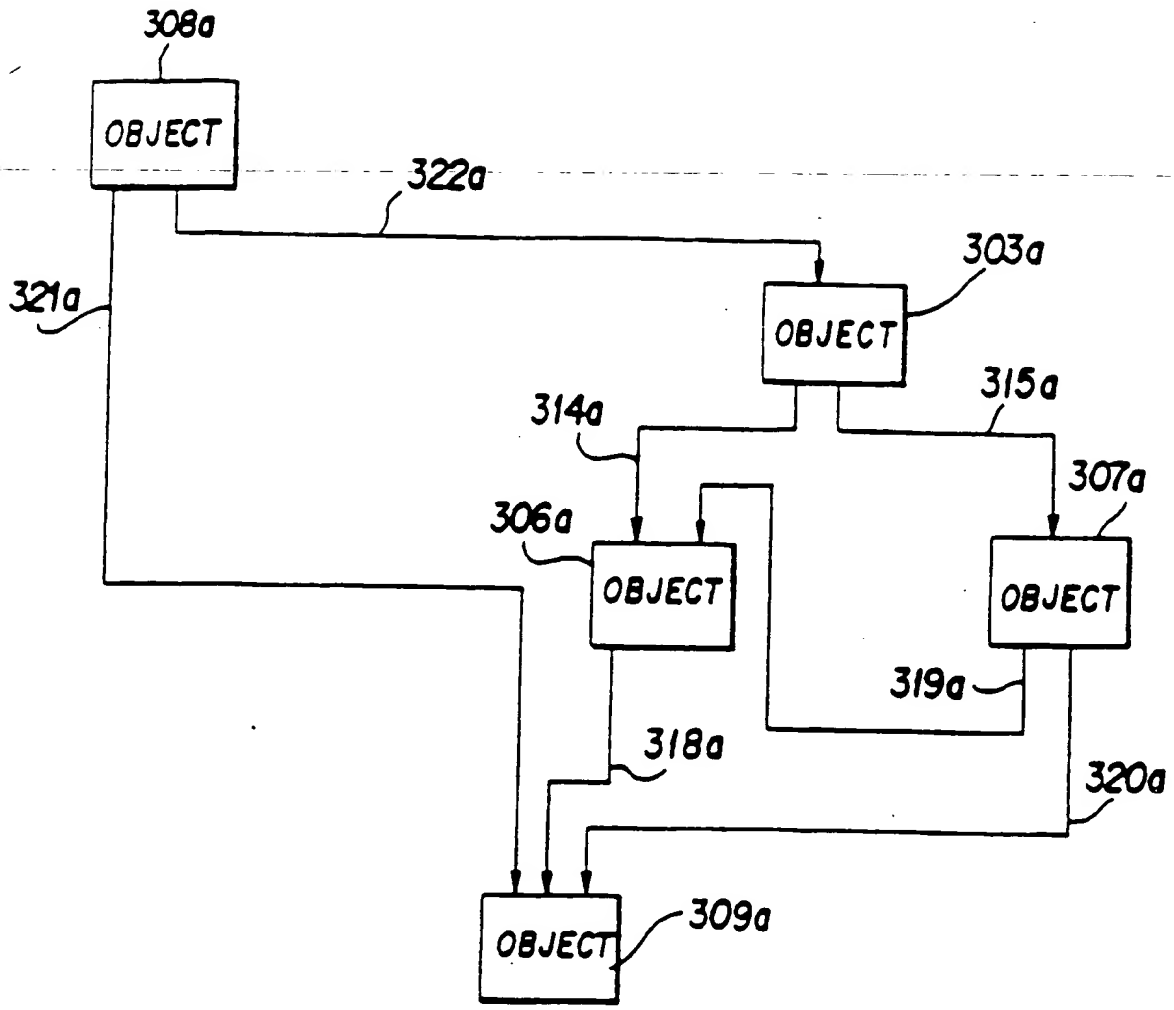

308a
OBJECT
322a
303a
321a
OBJECT
315a
314a
307a
306a
OBJECT
OBJECT
319a
318a
320a
OBJECT
309a

FIG. 9

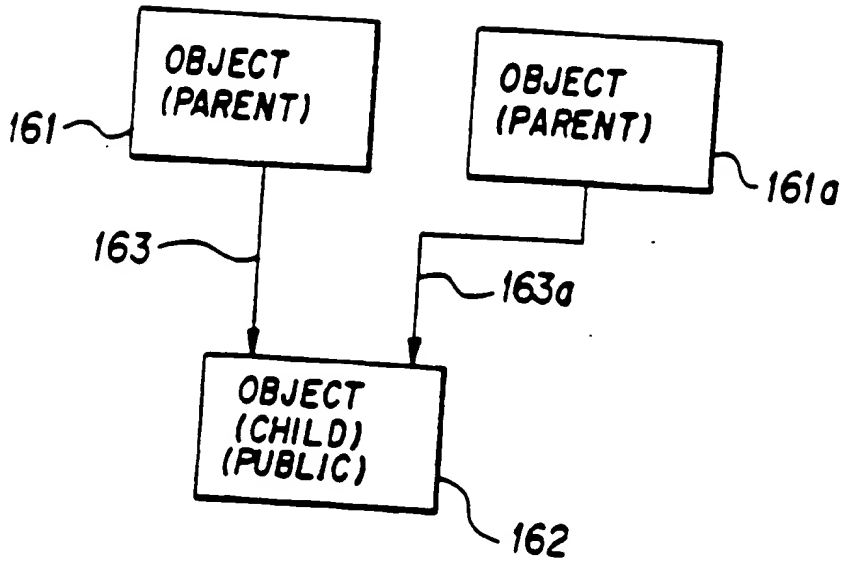

OBJECT
(PARENT)
161
OBJECT
(PARENT)
161a
163
163a
OBJECT
(CHILD)
(PUBLIC)
162

FIG. 11

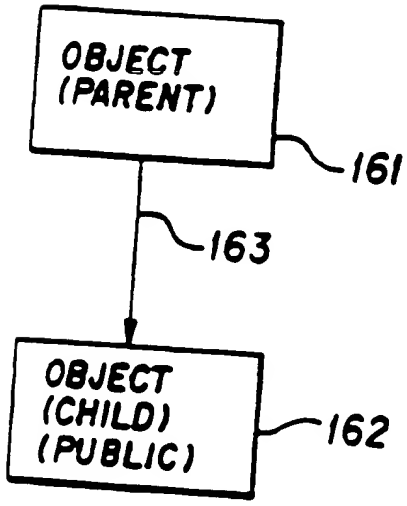

OBJECT
(PARENT)
161
163
OBJECT
(CHILD)
(PUBLIC)
162

FIG. 10

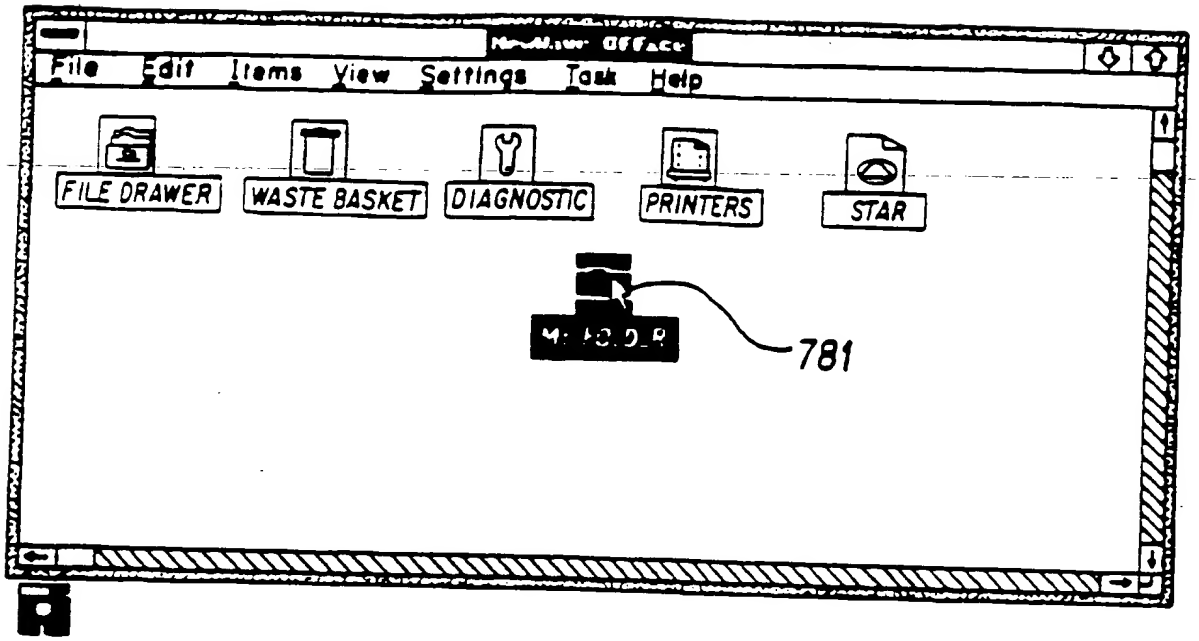

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
781

FIG. 12

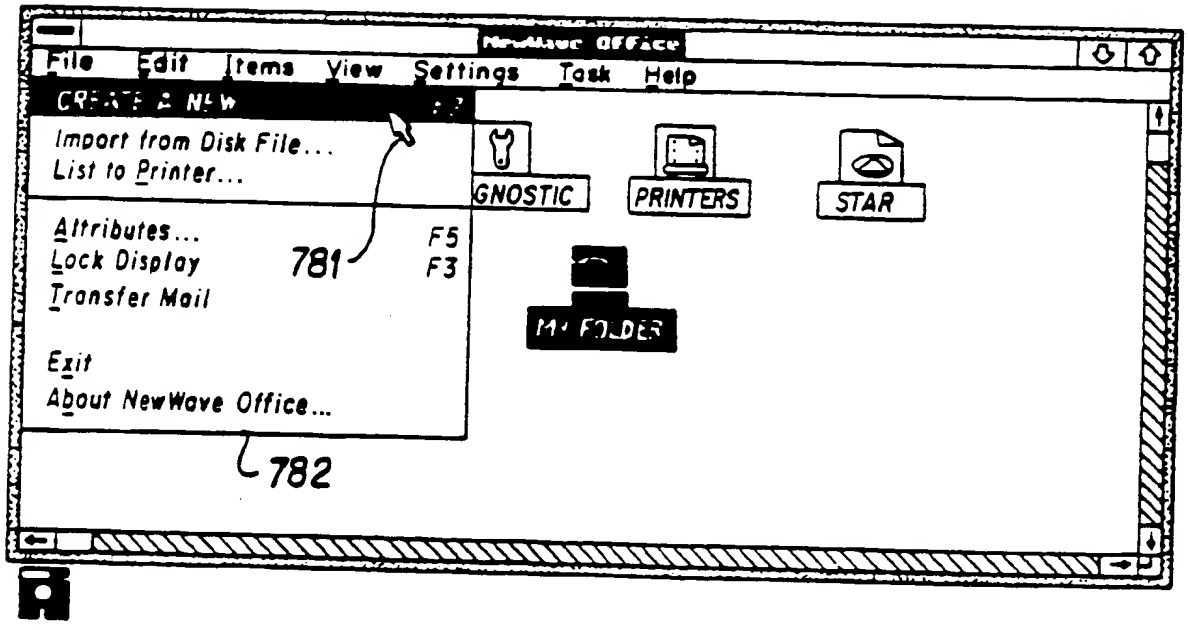

File Edit Items View Settings Task Help
Import from Disk File...
List to Printer...
Attributes... F5
Lock Display F3
Transfer Mail
Exit
About NewWave Office...
781
782
PRINTERS
STAR

FIG. 14

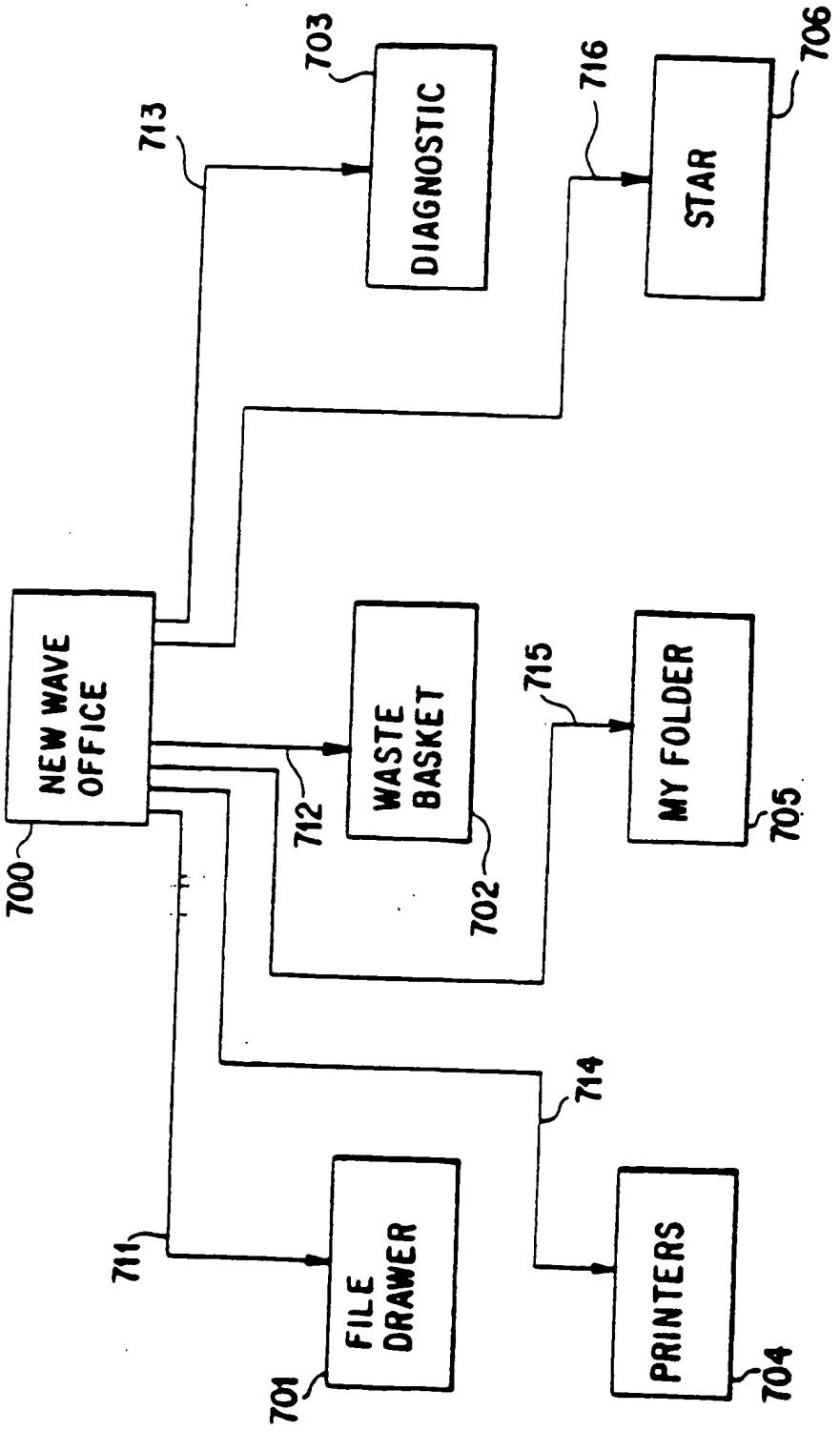

NEW WAVE OFFICE
700
DIAGNOSTIC
703
STAR
706
WASTE BASKET
702
MY FOLDER
705
FILE DRAWER
701
PRINTERS
704
711
712
713
714
715
716

FIG. 13

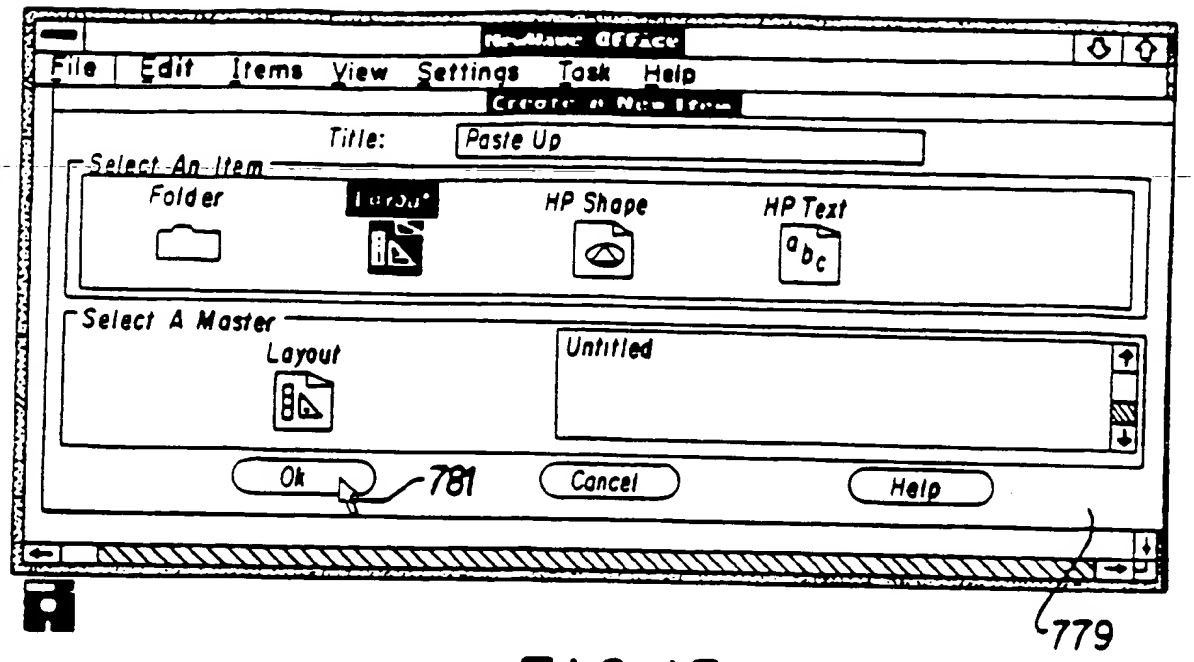

File Edit Items View Settings Task Help
Title:
Paste Up
Select An Item
Folder
HP Shape
HP Text
Select A Master
Layout
Untitled
Ok
Cancel
Help
781
779

FIG. 15

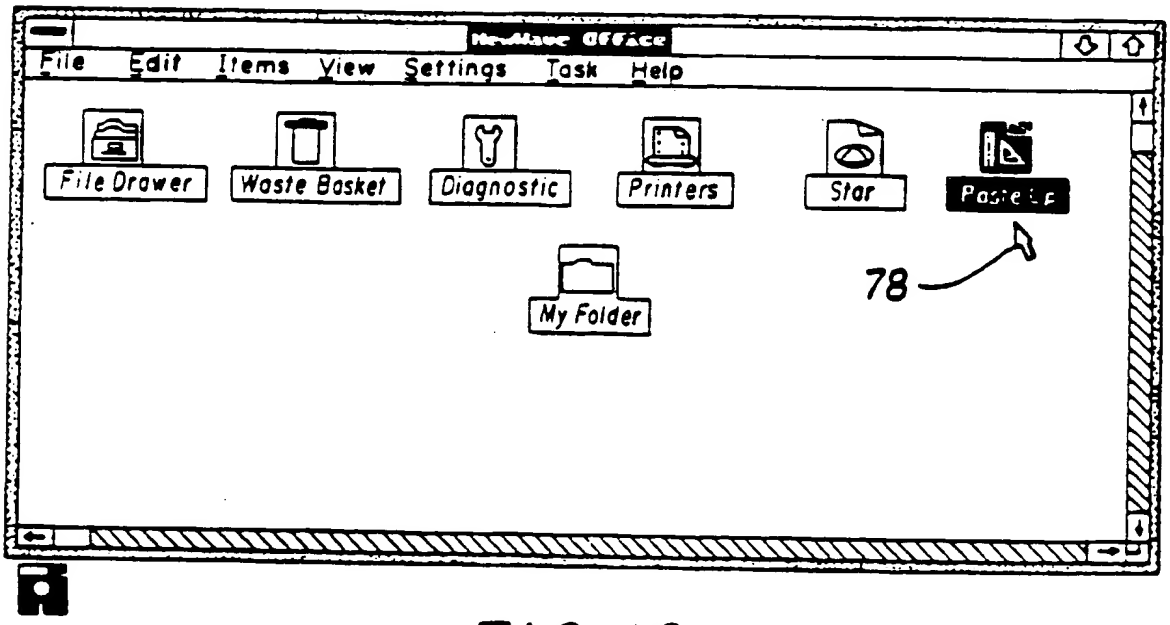

File Edit Items View Settings Task Help
File Drawer
Waste Basket
Diagnostic
Printers
Star
My Folder
78

FIG. 16

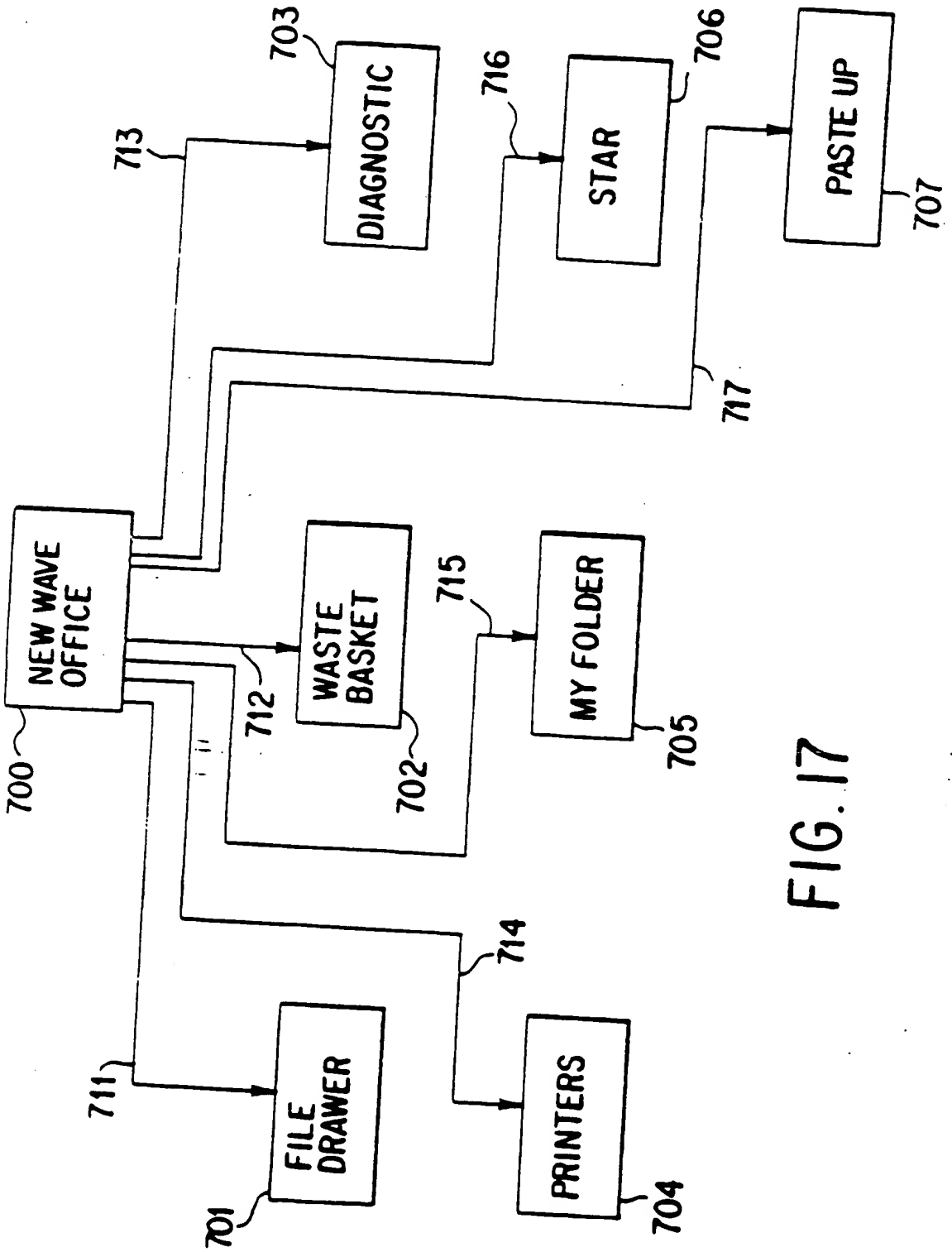

NEW WAVE OFFICE
700
DIAGNOSTIC
703
STAR
706
PASTE UP
707
WASTE BASKET
702
MY FOLDER
705
FILE DRAWER
701
PRINTERS
704
711
712
713
714
715
716
717

FIG. 17

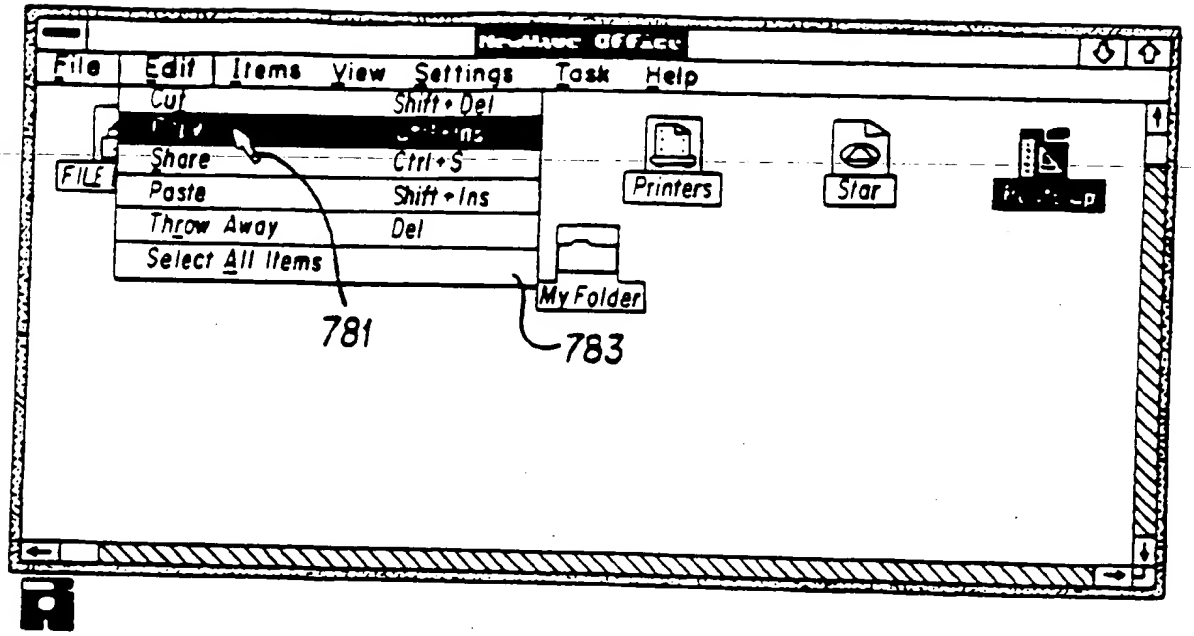

File
Edit
Items
View
Settings
Task
Help
Cut
Shift + Del
Share
Ctrl + S
Paste
Shift + Ins
Throw Away
Del
Select All Items
Printers
Star
My Folder
781
783

FIG. 18

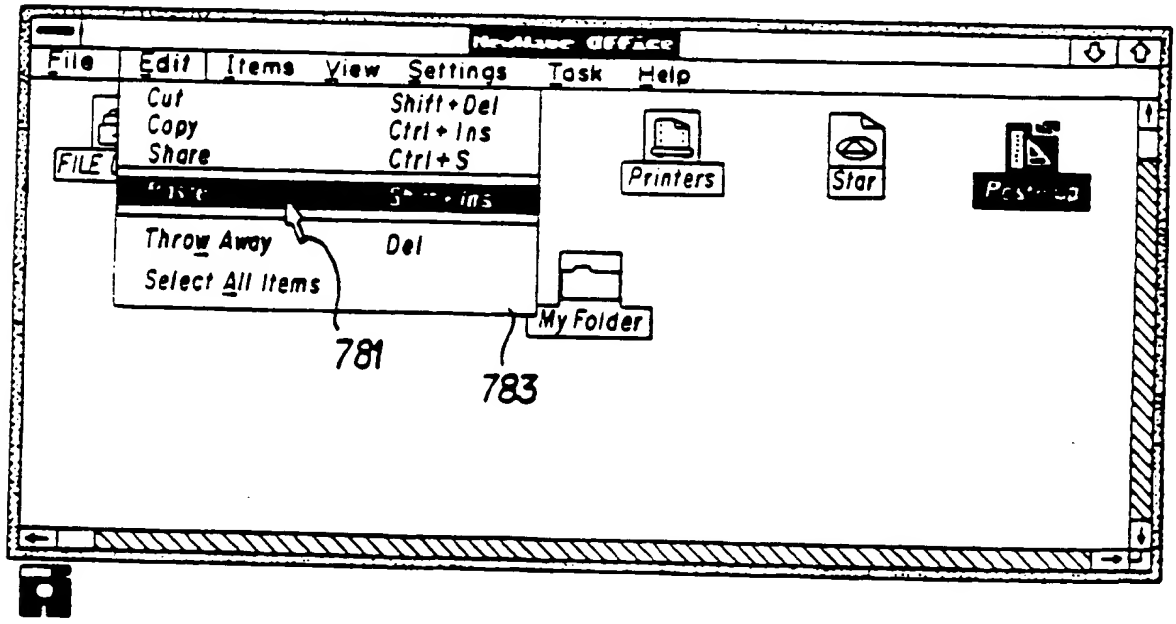

File
Edit
Items
View
Settings
Task
Help
Cut
Shift + Del
Copy
Ctrl + Ins
Share
Ctrl + S
Throw Away
Del
Select All Items
Printers
Star
My Folder
781
783

FIG. 19

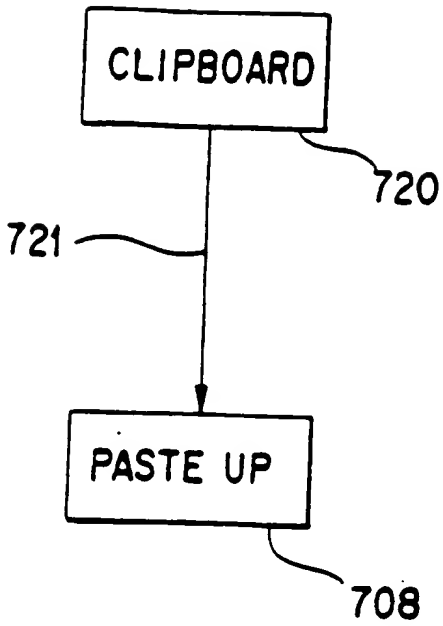


FIG. 18A

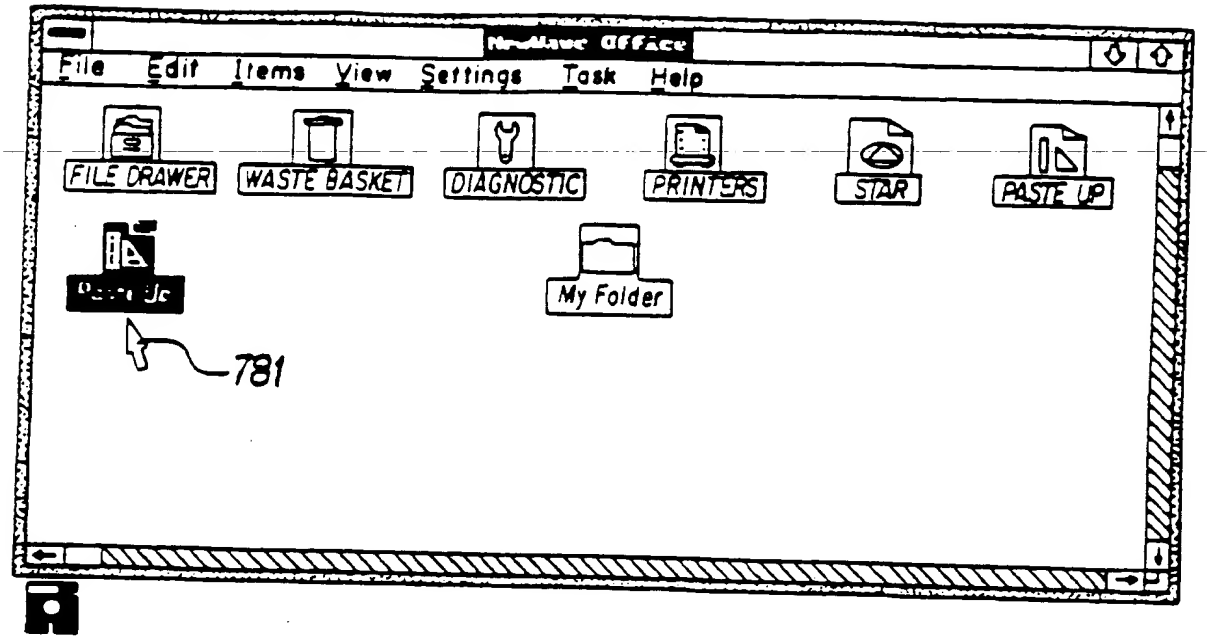

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
My Folder
781

FIG. 20

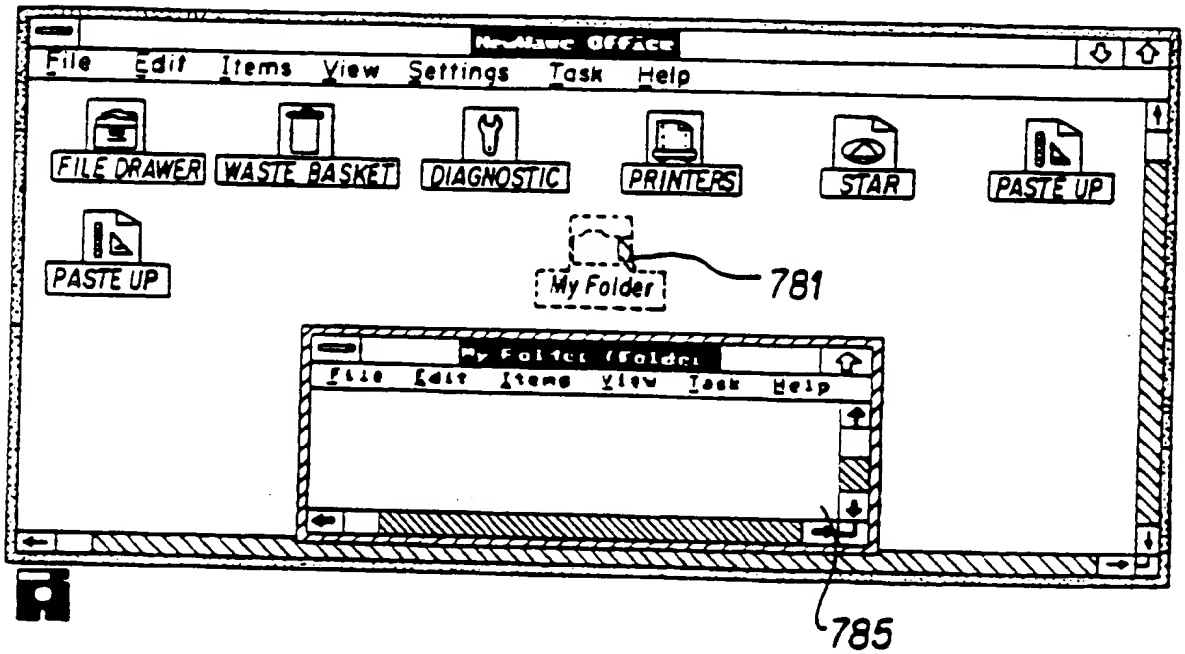

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
PASTE UP
My Folder
781
File Edit Items View Task Help
785

FIG. 22

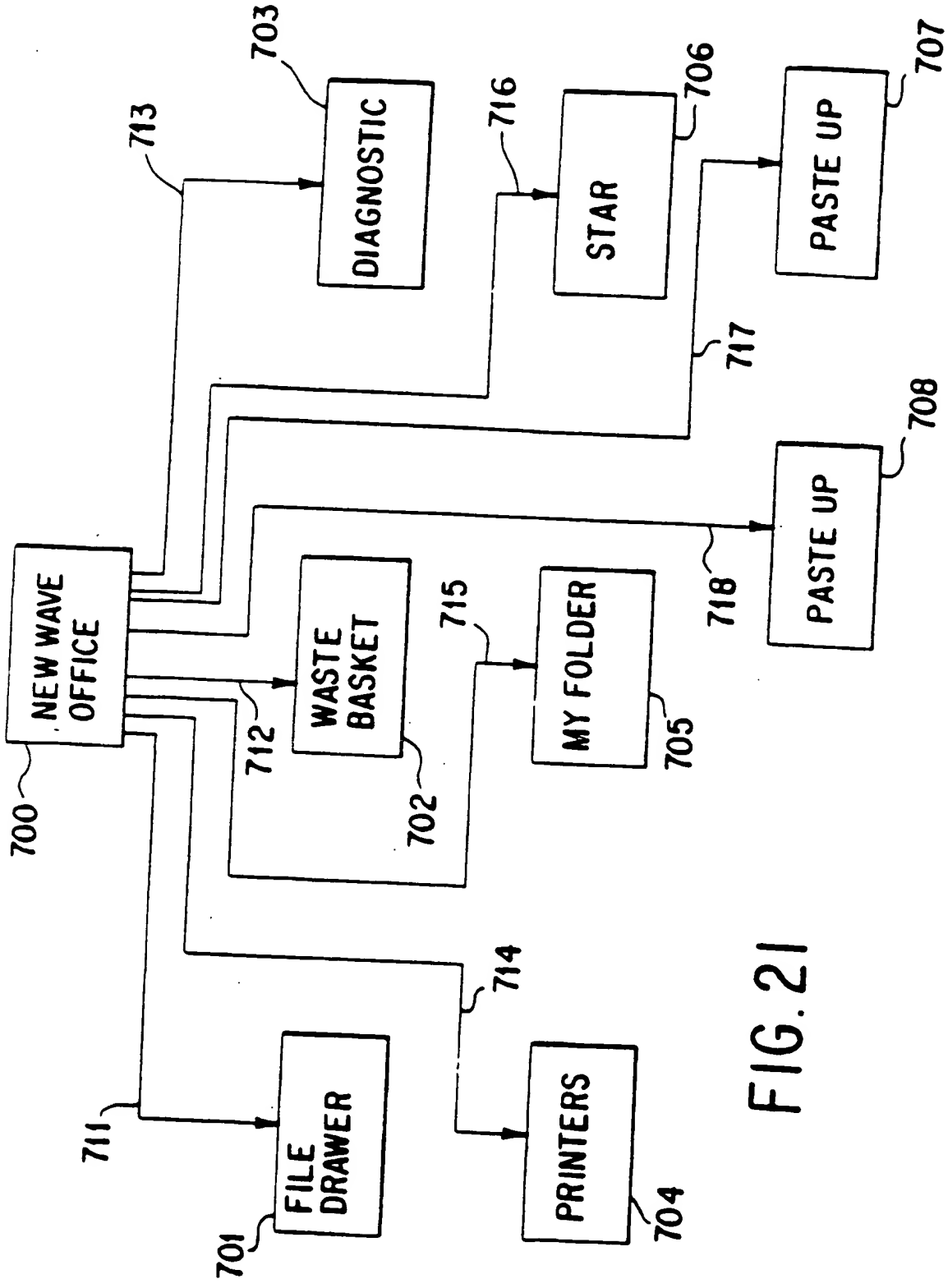

NEW WAVE OFFICE
700
DIAGNOSTIC
703
STAR
706
PASTE UP
707
PASTE UP
708
WASTE BASKET
702
MY FOLDER
705
FILE DRAWER
701
PRINTERS
704
711
712
713
714
715
716
717
718

FIG. 21

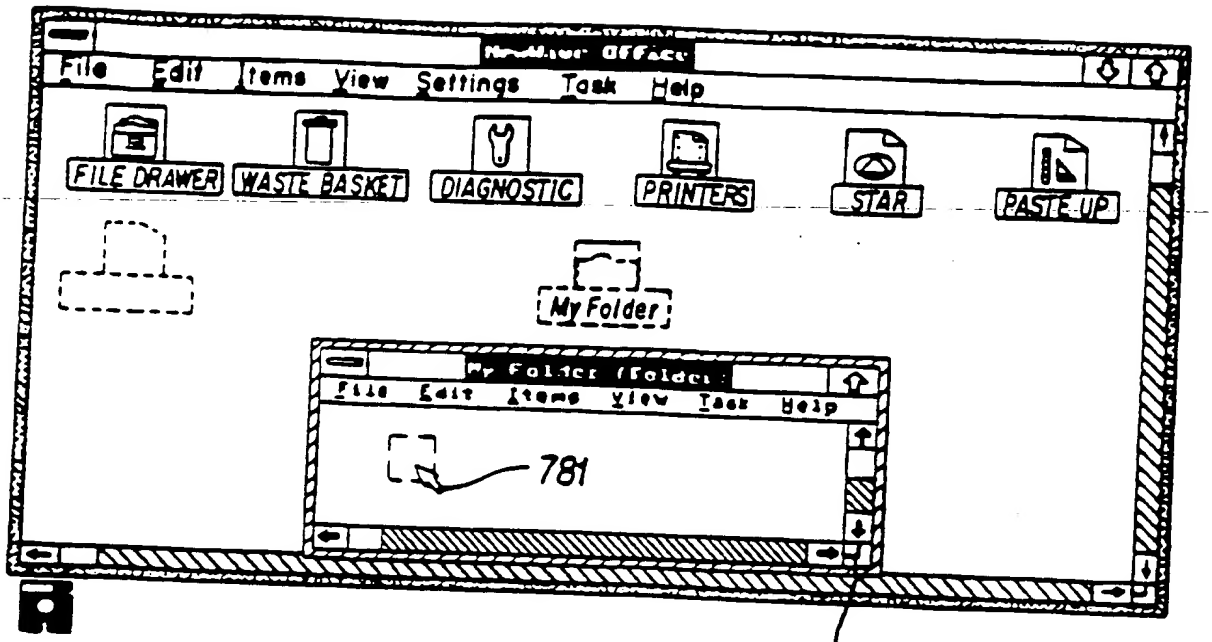

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
My Folder
File Edit Items View Task Help
781
785

FIG. 23

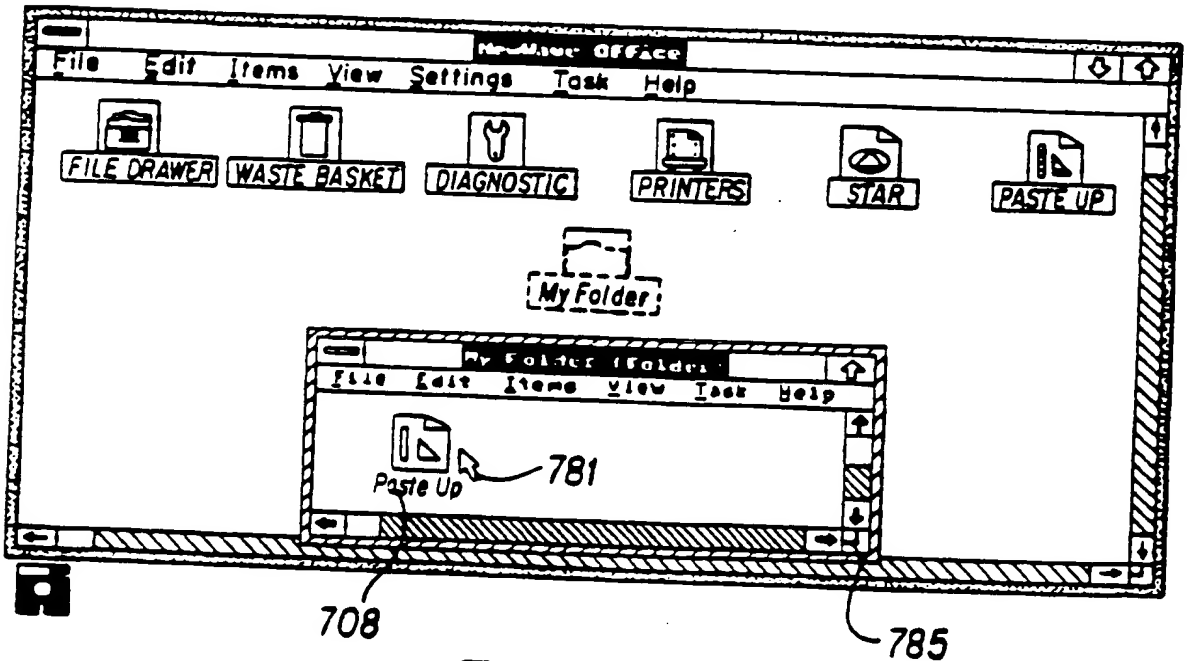

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
My Folder
File Edit Items View Task Help
Paste Up
781
708
785

FIG. 24

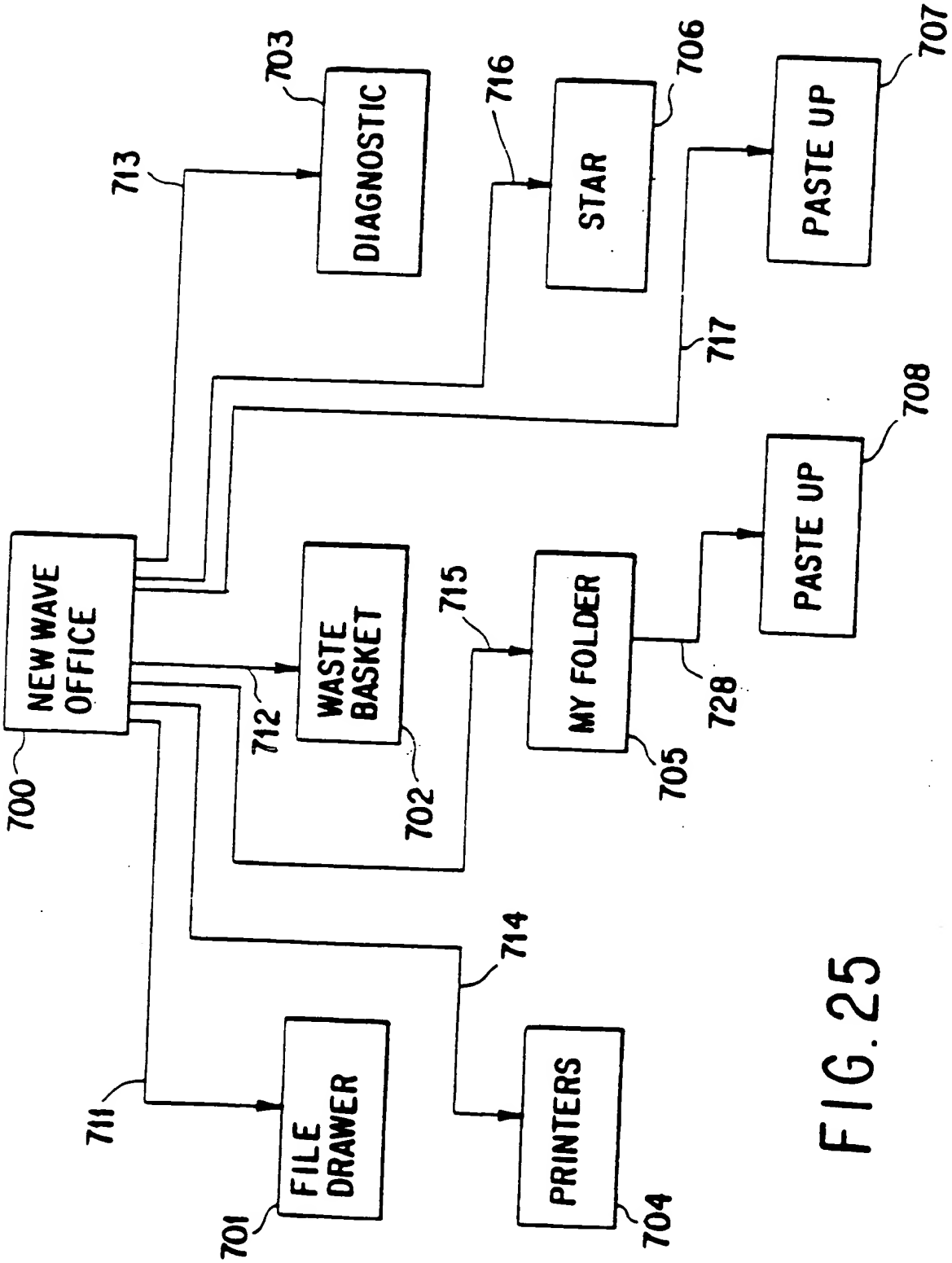

NEW WAVE OFFICE
700
DIAGNOSTIC
703
STAR
706
PASTE UP
707
PASTE UP
708
WASTE BASKET
702
MY FOLDER
705
FILE DRAWER
701
PRINTERS
704
711
712
713
714
715
716
717
728

FIG. 25

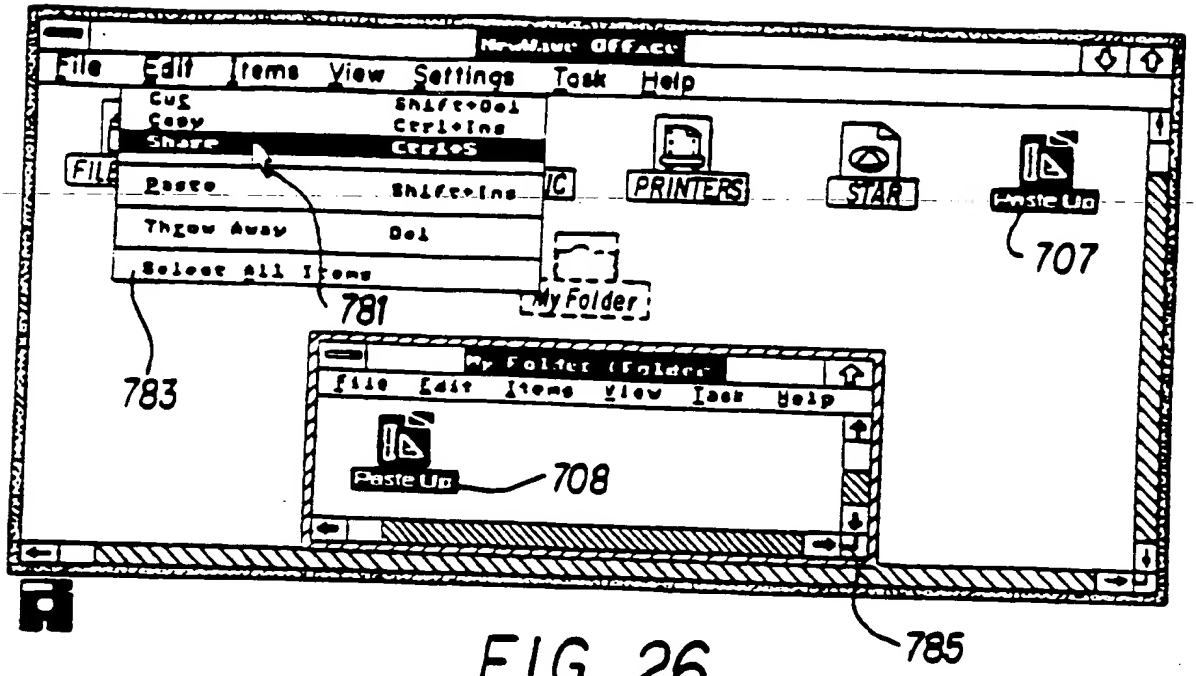

File Edit Items View Settings Task Help
Cut Shift+Del
Copy Ctrl+Ins
Share Ctrl+S
Paste Shift+Ins
Throw Away Del
Select All Items
781
783
PRINTERS
STAR
Paste Up
707
My Folder
My Folder (Folder
File Edit Items View Task Help
Paste Up
708
785

FIG. 26

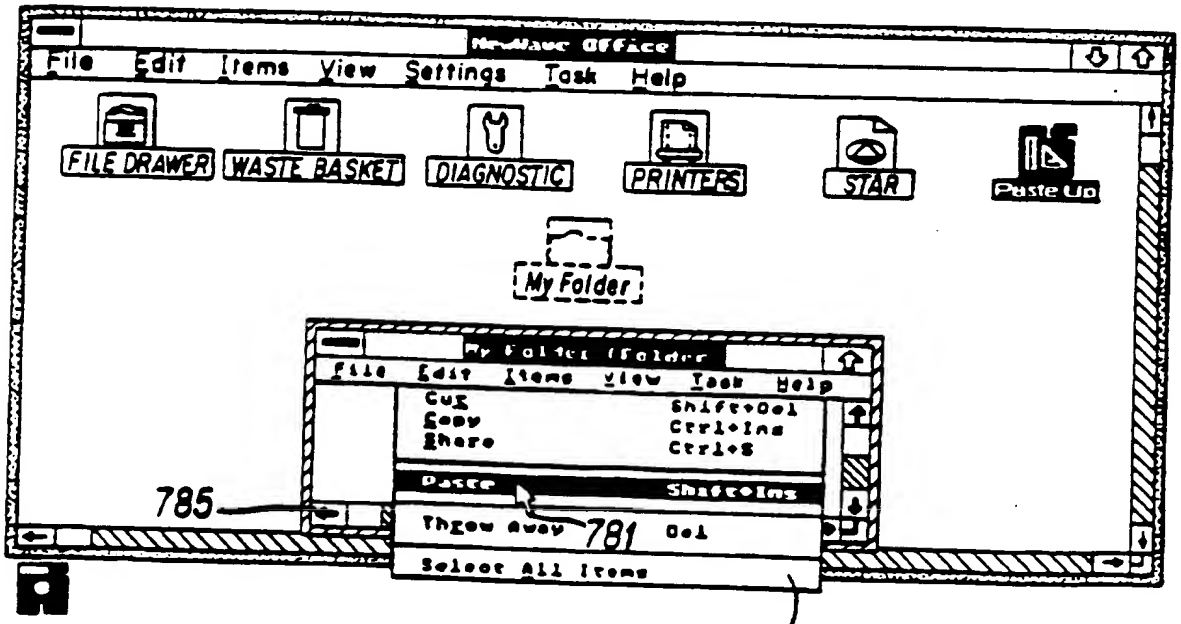

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
Paste Up
My Folder
My Folder (Folder
File Edit Items View Task Help
Cut Shift+Del
Copy Ctrl+Ins
Share Ctrl+S
Paste Shift+Ins
Throw Away Del
Select All Items
785
781
787

FIG. 27

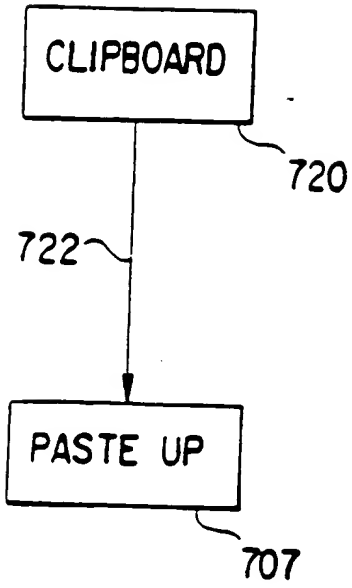


FIG. 26A

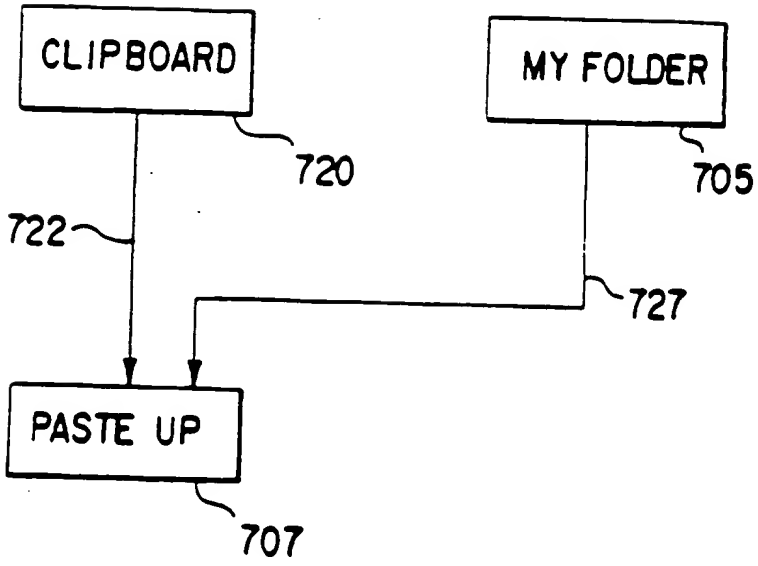


FIG. 28A

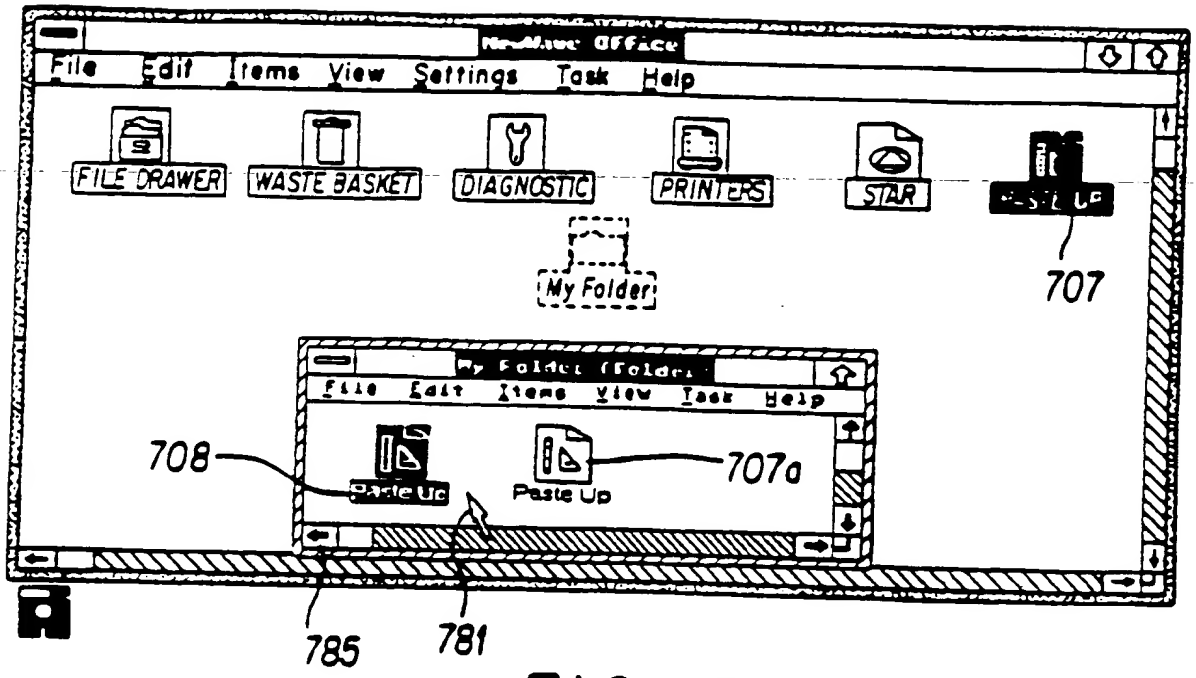

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
My Folder
707
File Edit Items View Task Help
708
Paste Up
707a
785
781

FIG. 28

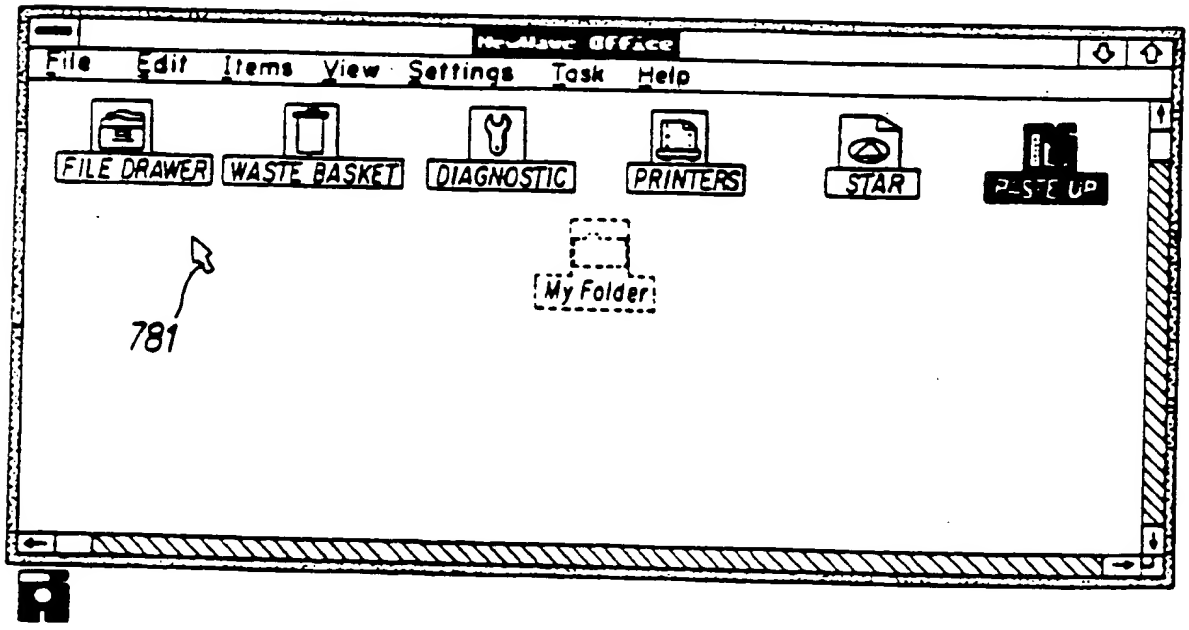

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
My Folder
781

FIG. 30

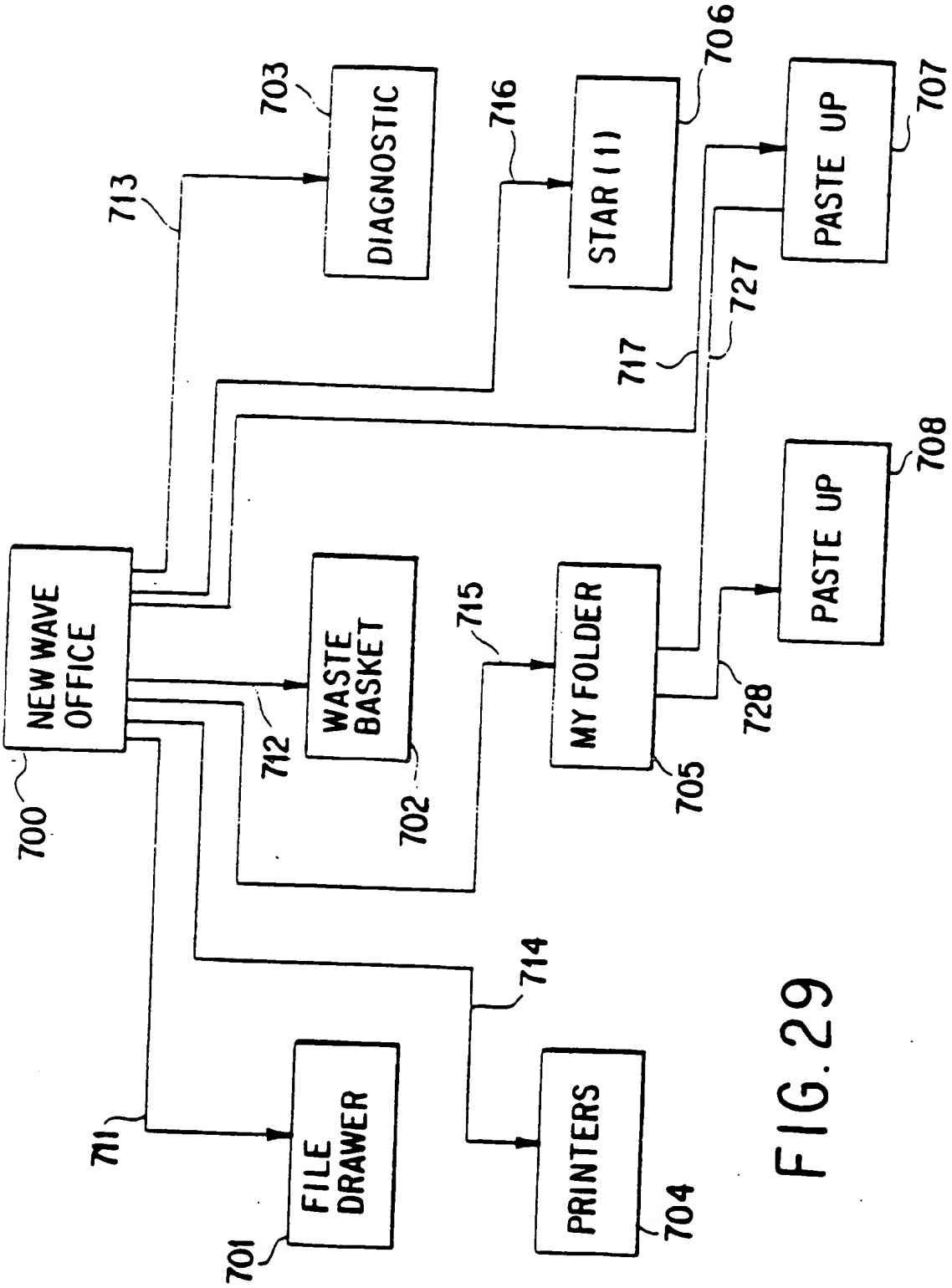

NEW WAVE OFFICE
700
DIAGNOSTIC
703
STAR (1)
706
PASTE UP
707
PASTE UP
708
WASTE BASKET
702
MY FOLDER
705
FILE DRAWER
701
PRINTERS
704
711
712
713
714
715
716
717
727
728

FIG. 29

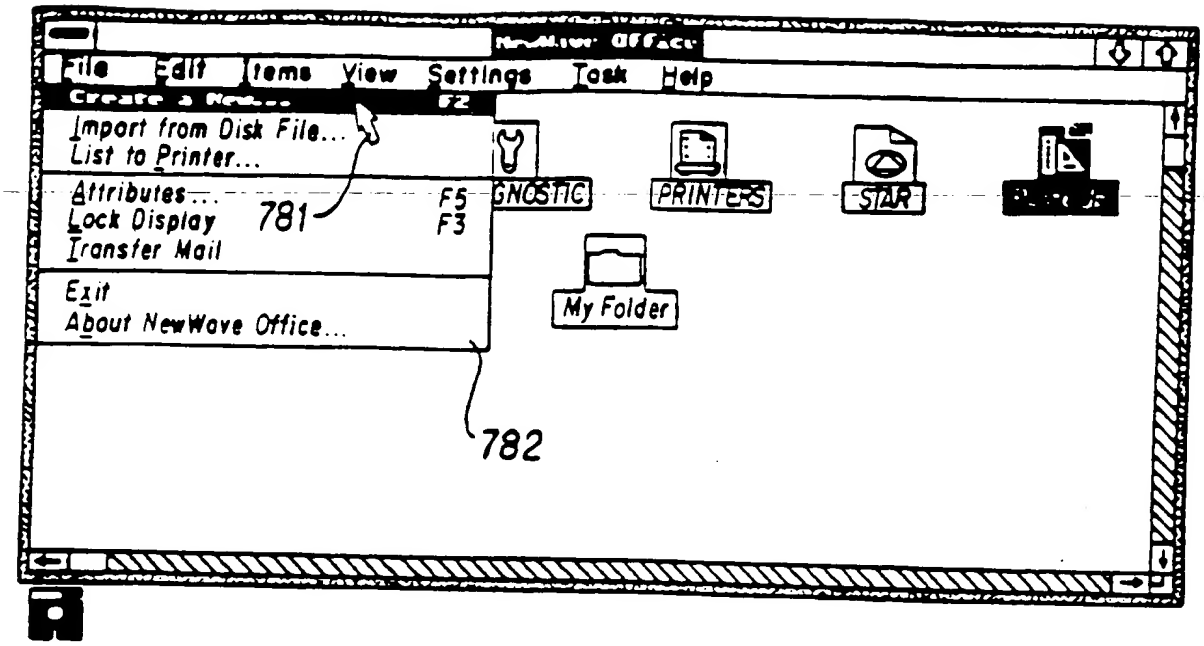

File Edit Items View Settings Task Help
Create a New... F2
Import from Disk File...
List to Printer...
Attributes... F5
Lock Display F3
Transfer Mail
Exit
About NewWave Office...
781
782
PRINTERS
My Folder

FIG. 31

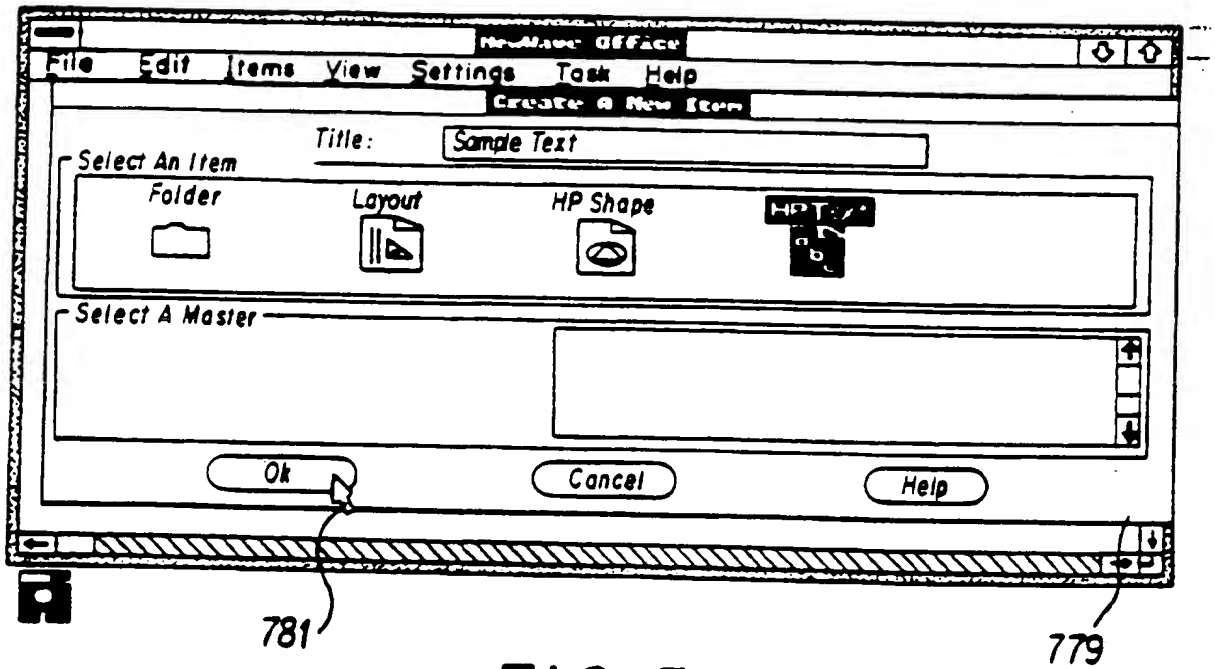

File Edit Items View Settings Task Help
Create a New Item
Title:
Sample Text
Select An Item
Folder
Layout
HP Shape
Select A Master
Ok
Cancel
Help
781
779

FIG. 32

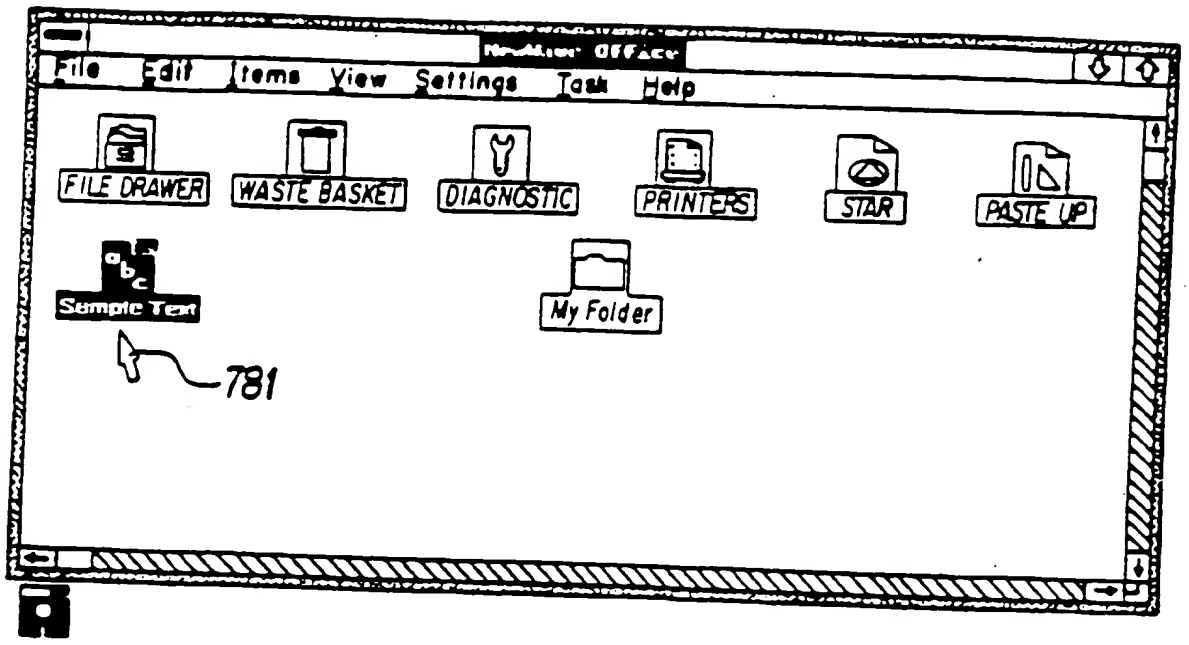

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
Sample Text
My Folder
781

FIG. 33

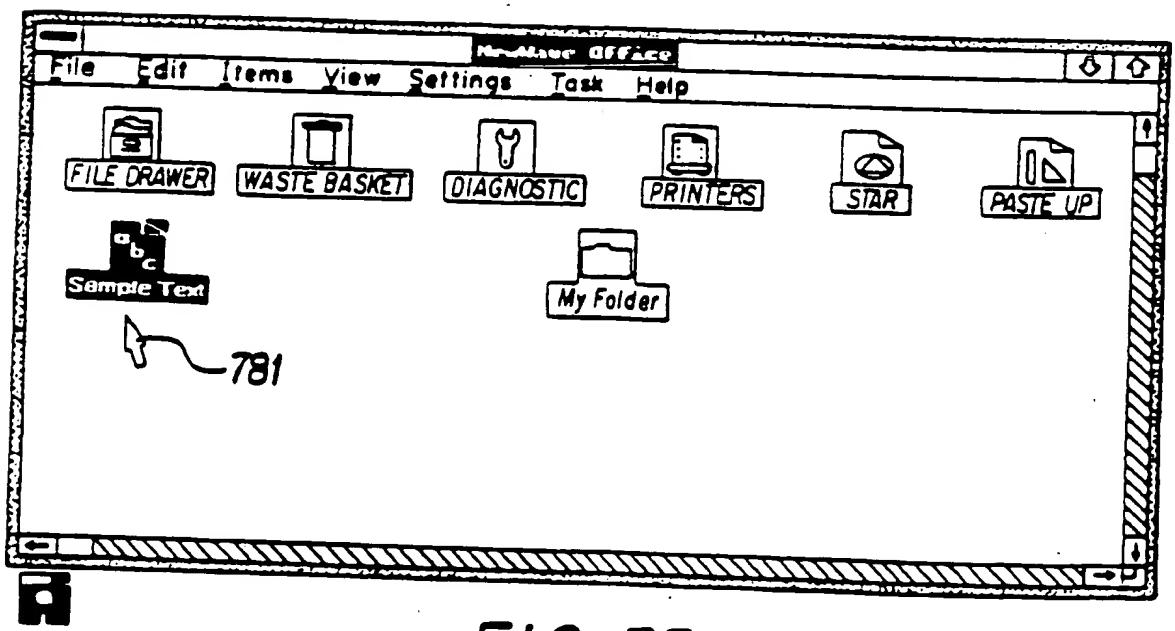

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
Sample Text
My Folder
781

FIG. 35

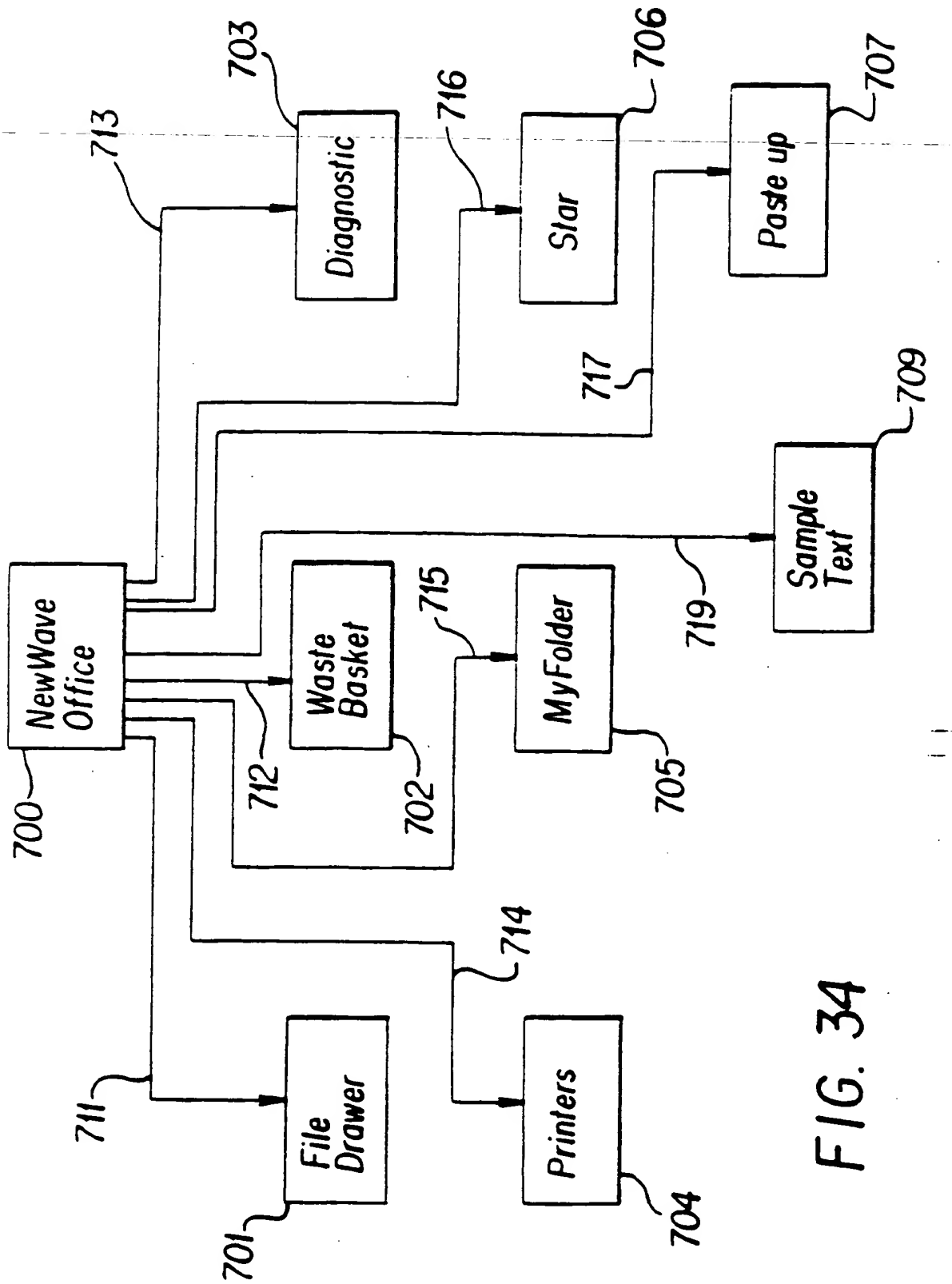

NewWave Office
700
Diagnostic
703
Star
706
Paste up
707
Waste Basket
712
MyFolder
715
Sample Text
709
File Drawer
701
Printers
704
713
716
717
719
711
714
702
705

FIG. 34

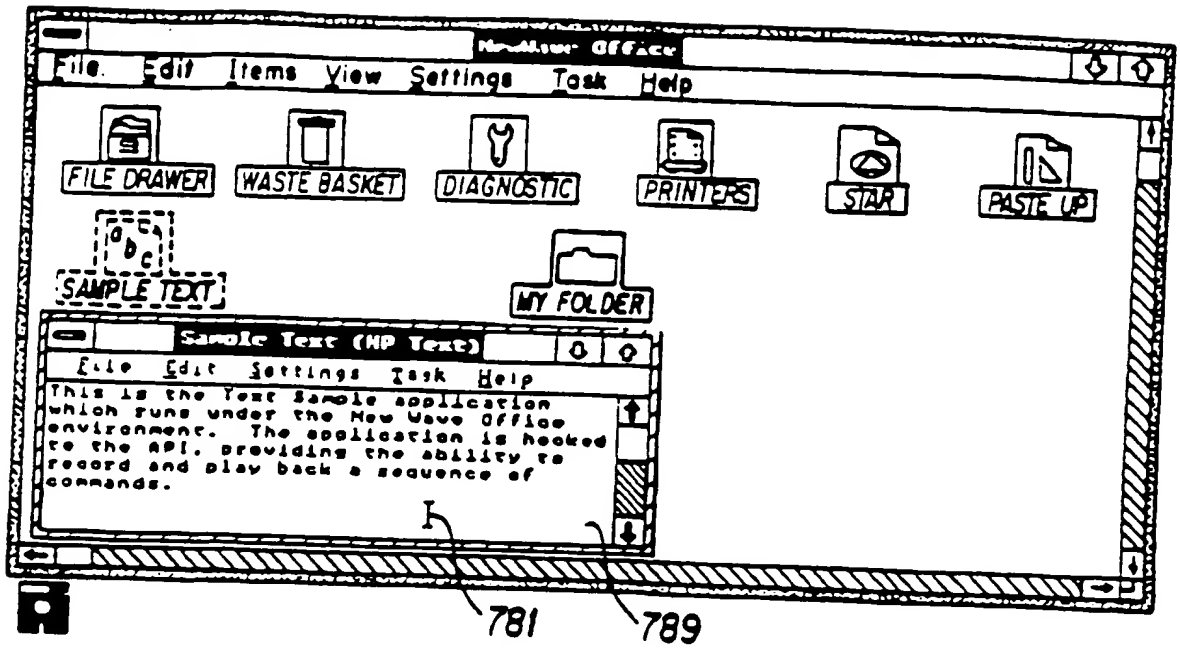

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
SAMPLE TEXT
MY FOLDER
Sample Text (HP Text)
File Edit Settings Task Help
This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.
781
789

FIG. 36

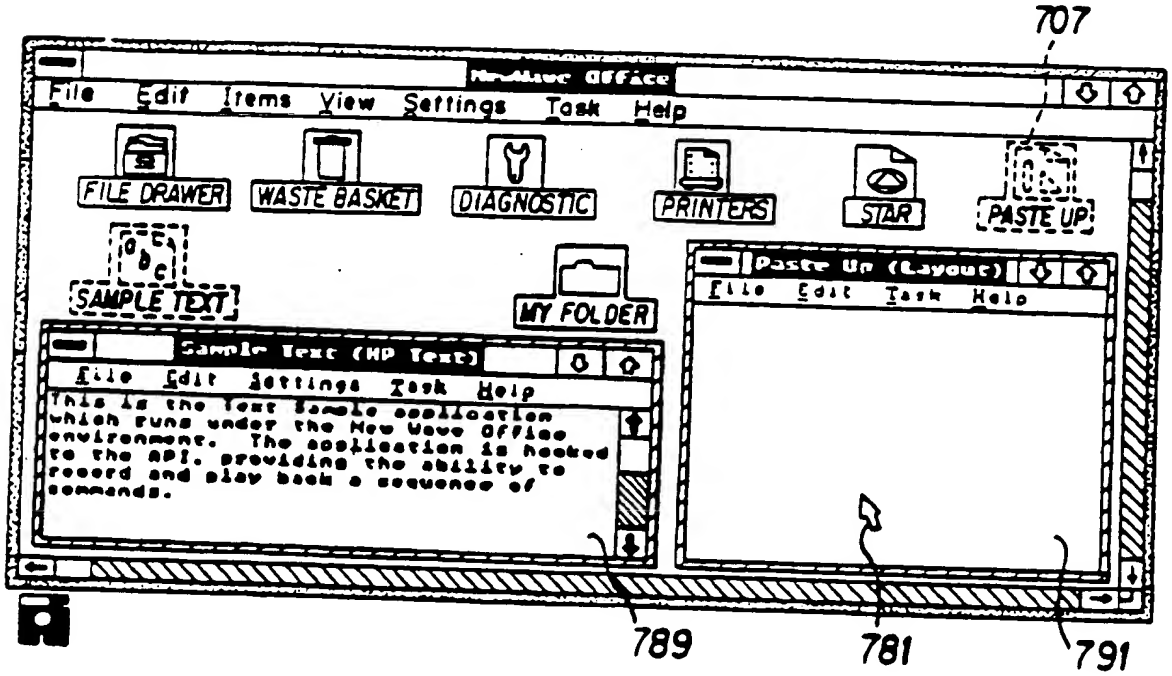

707
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
SAMPLE TEXT
MY FOLDER
Sample Text (HP Text)
File Edit Settings Task Help
This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.
Paste Up (Layout)
File Edit Task Help
789
781
791

FIG. 37

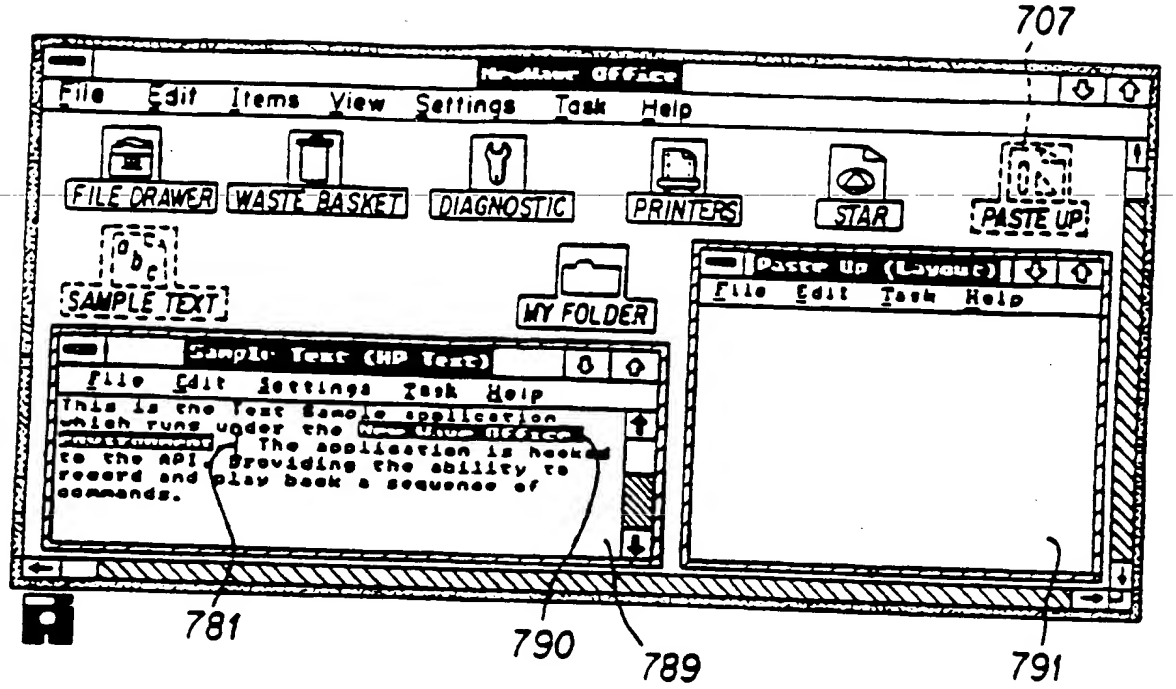

707
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
SAMPLE TEXT
MY FOLDER
Sample Text (HP Text)
File Edit Settings Task Help
Paste Up (Layout)
File Edit Task Help
781
790
789
791

FIG. 38

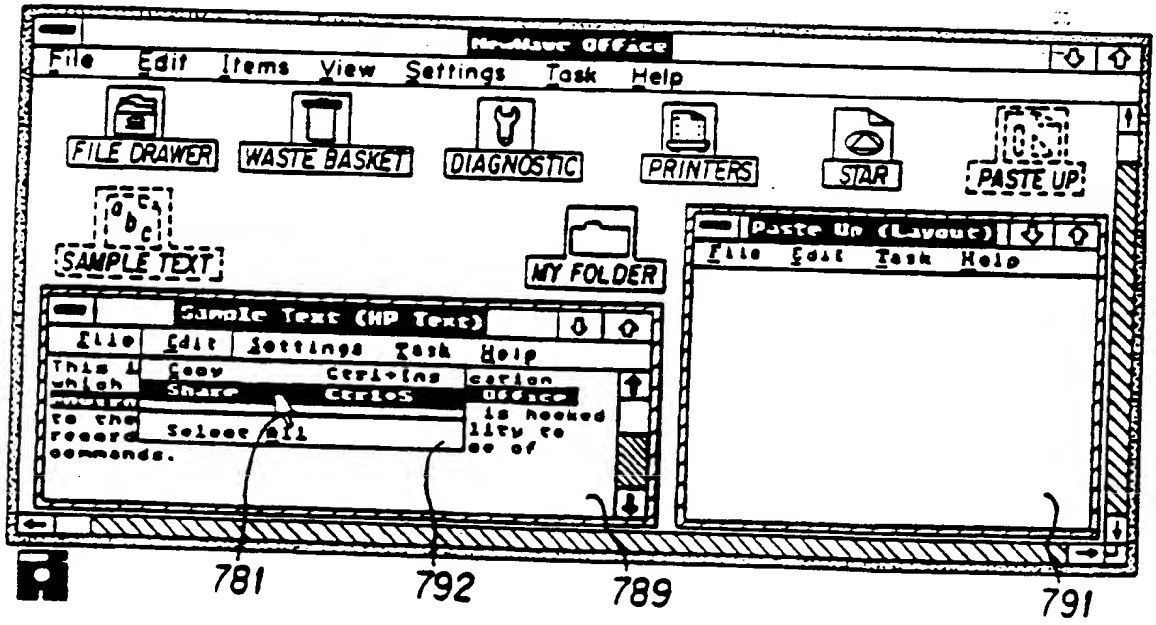

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
SAMPLE TEXT
MY FOLDER
Sample Text (HP Text)
File Edit Settings Task Help
Paste Up (Layout)
File Edit Task Help
781
792
789
791

FIG. 39

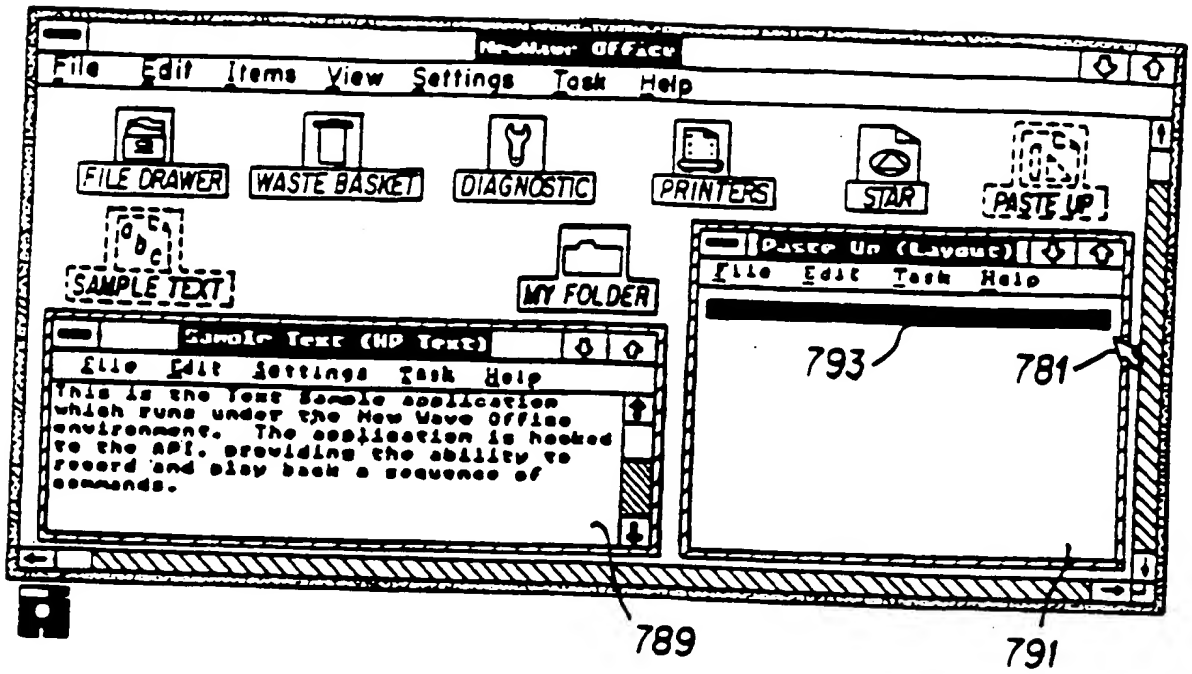

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
SAMPLE TEXT
MY FOLDER
Paste Up (Layout)
File Edit Task Help
Sample Text (MP Text)
File Edit Settings Task Help
This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.
793
781
789
791

FIG.40

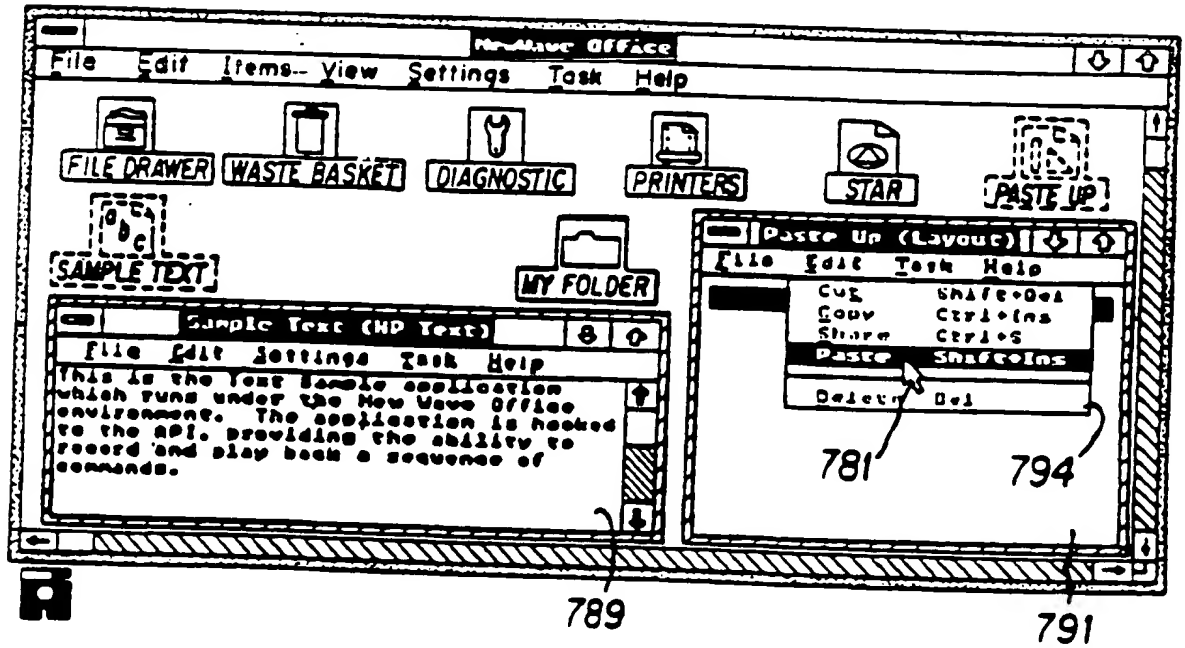

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
PASTE UP
SAMPLE TEXT
MY FOLDER
Paste Up (Layout)
File Edit Task Help
Cut Shift+Del
Copy Ctrl+Ins
Paste Shift+Ins
Delete Del
Sample Text (MP Text)
File Edit Settings Task Help
This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.
781
794
789
791

FIG. 41

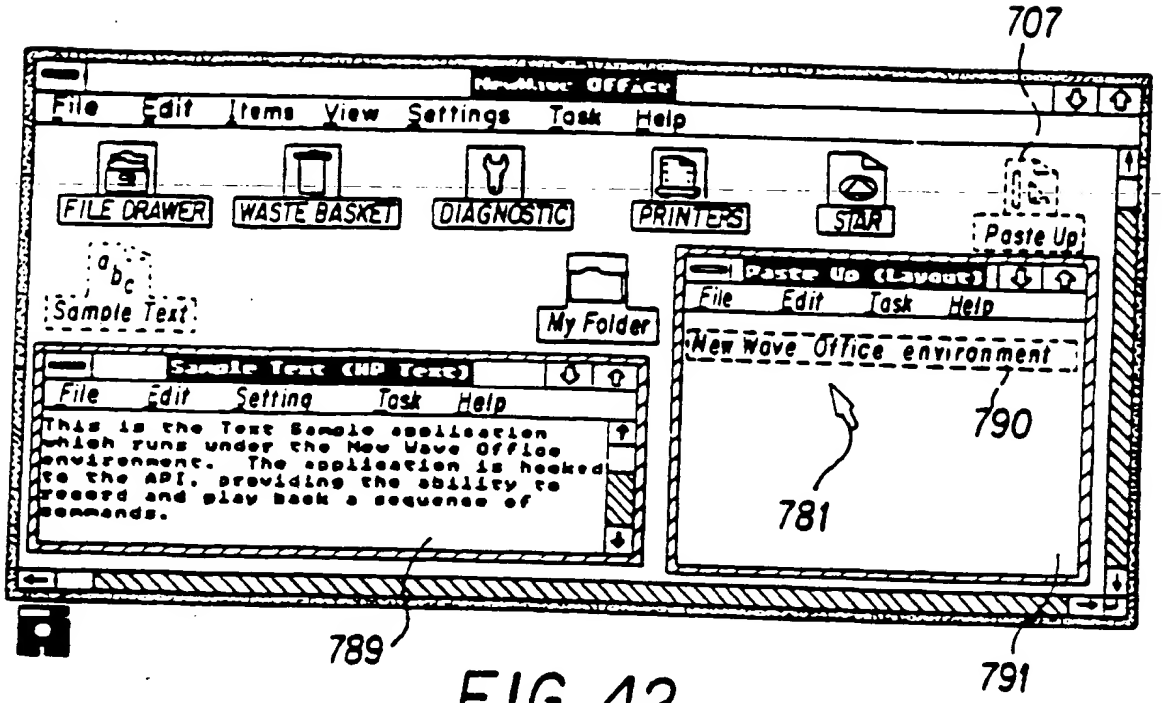

707
File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
790
781
Sample Text (MP Text)
File Edit Setting Task Help
This is the Text Sample application which runs under the New Wave Office environment. The application is hooked to the API, providing the ability to record and play back a sequence of commands.
789
791

FIG. 42

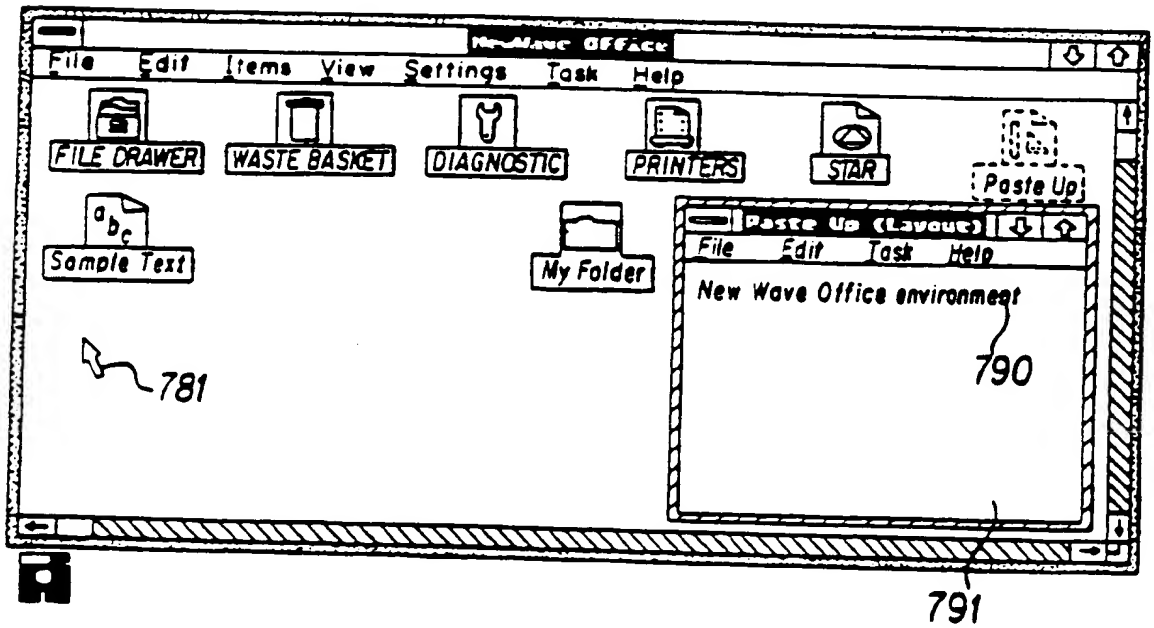

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
790
781
791

FIG. 44

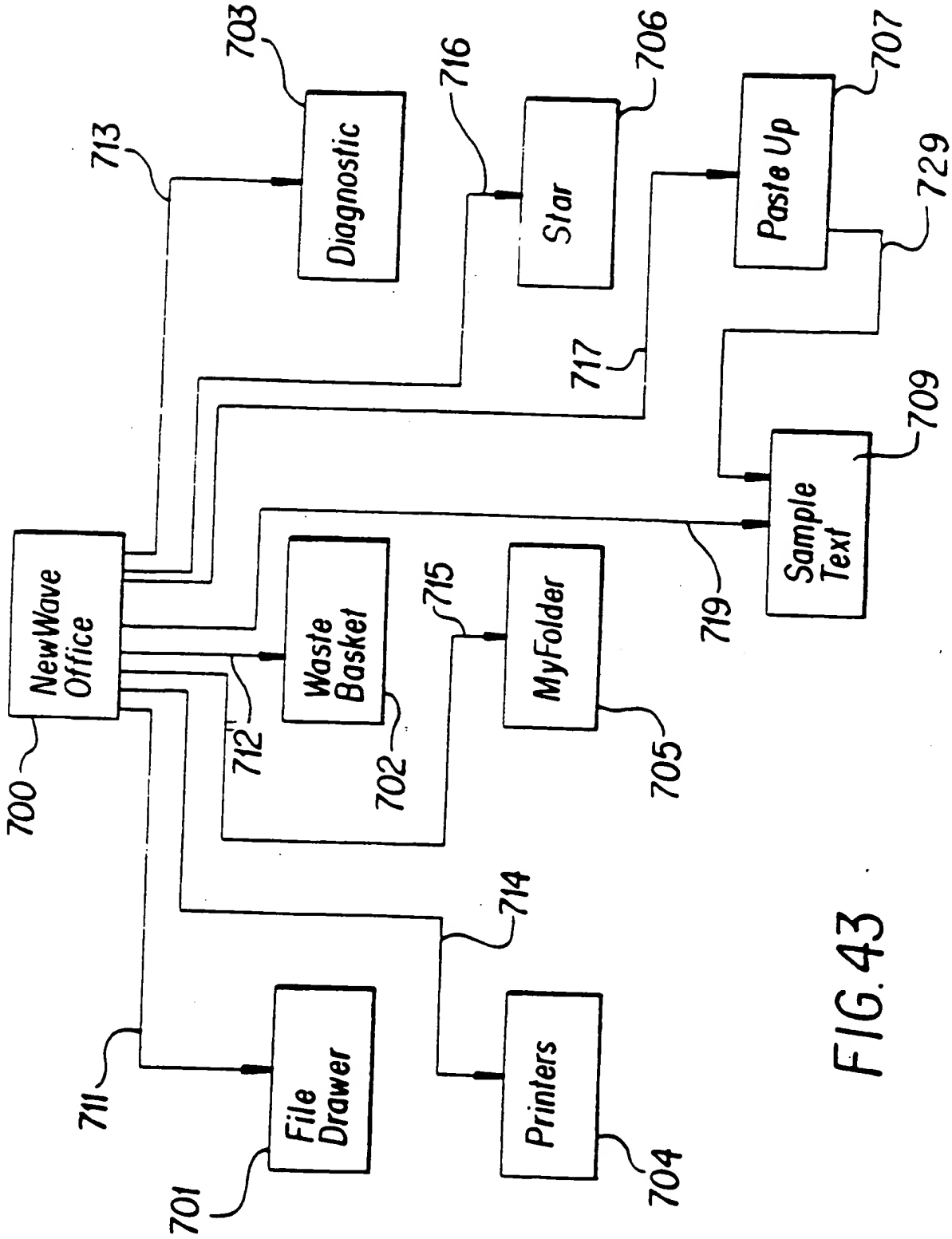

NewWave Office
700
File Drawer
701
Waste Basket
702
Printers
704
MyFolder
705
Star
706
Paste Up
707
Sample Text
709
Diagnostic
703
711
712
713
714
715
716
717
719
729

FIG. 43

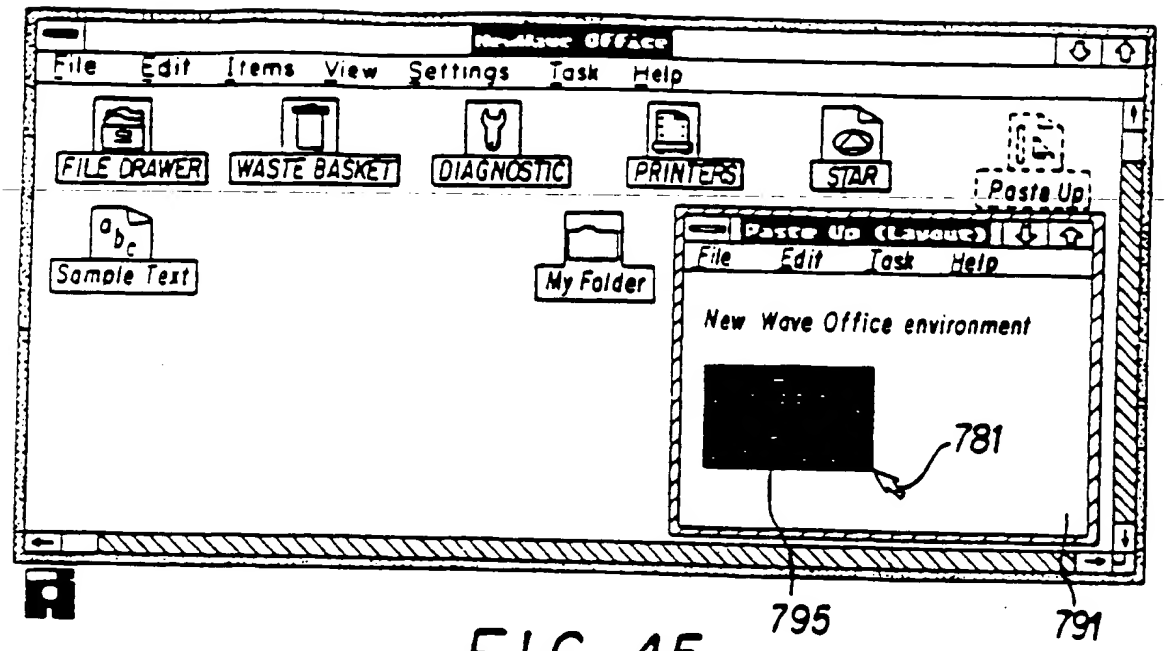

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
STAR
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
781
795
791

FIG. 45

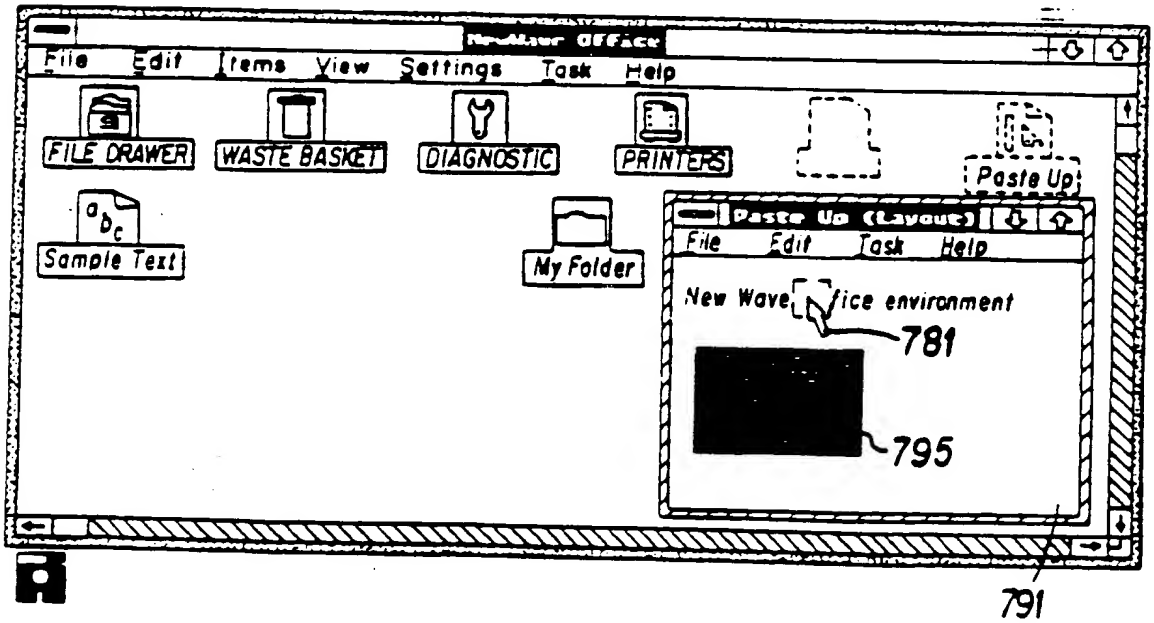

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
781
795
791

FIG. 46

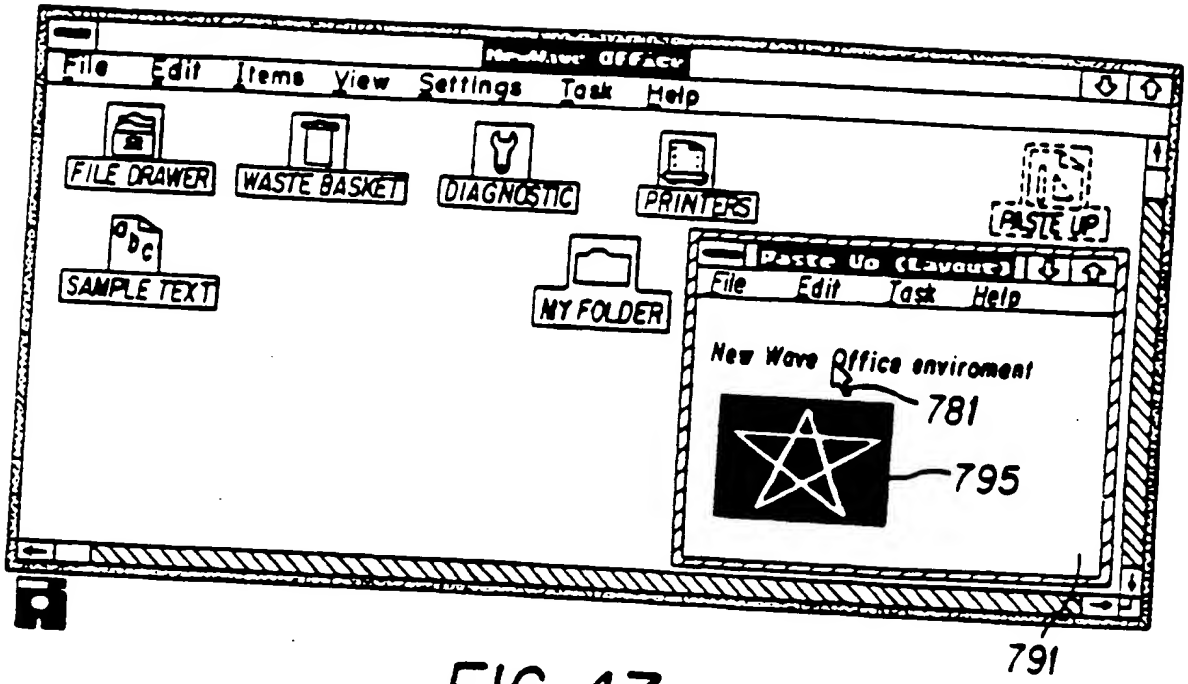

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
781
795
791

FIG. 47

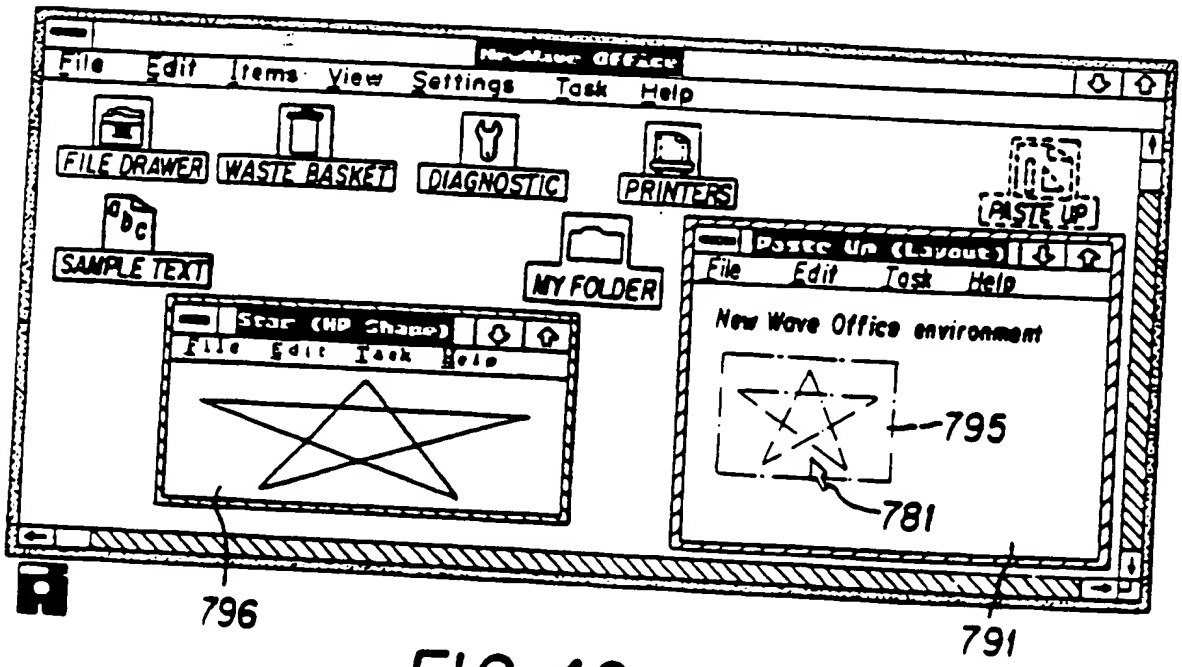

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
795
781
796
791

FIG. 49

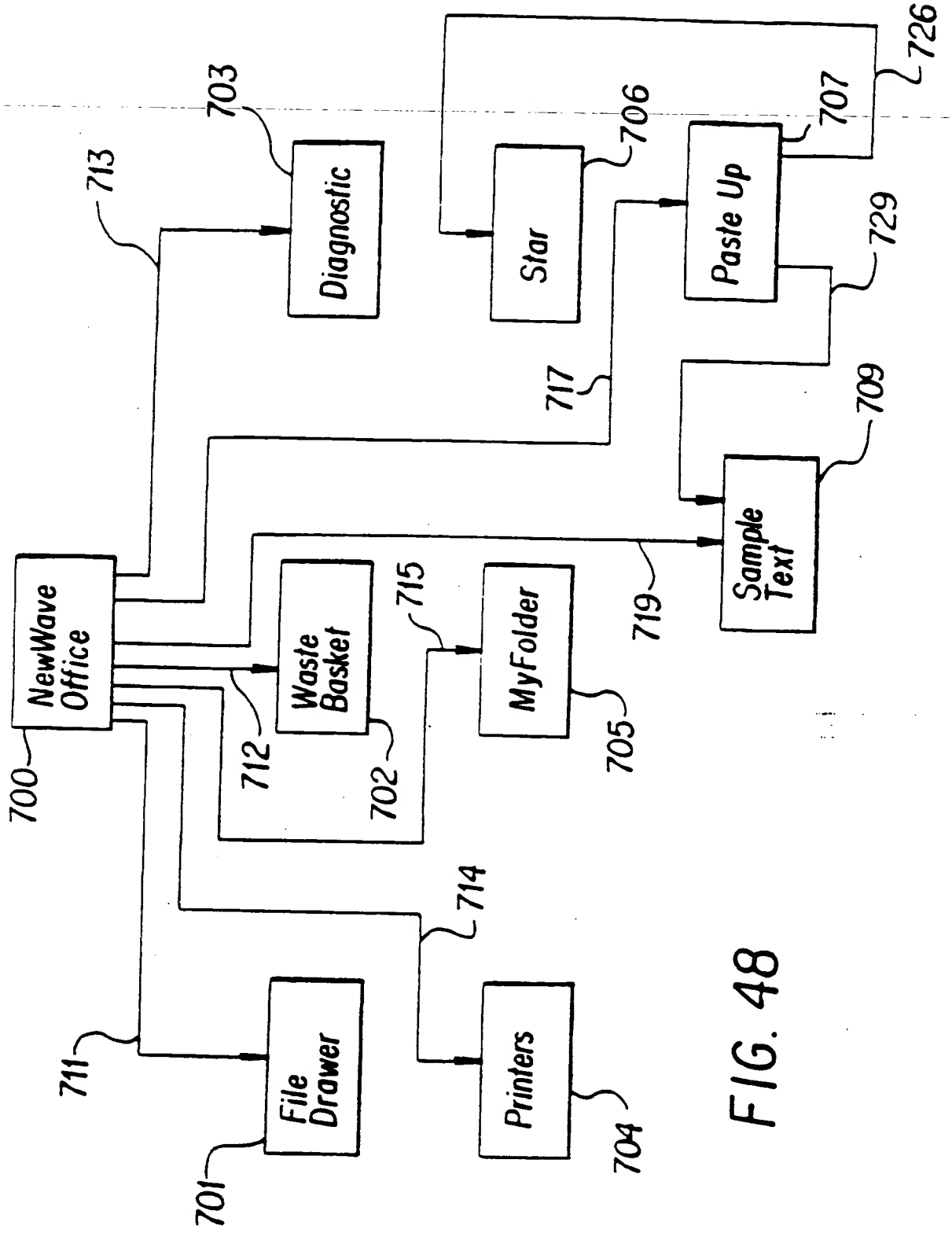

NewWave Office
700
Diagnostic
703
Star
706
Paste Up
707
726
729
Sample Text
709
719
717
713
Waste Basket
712
MyFolder
715
705
702
File Drawer
711
701
Printers
714
704

FIG. 48

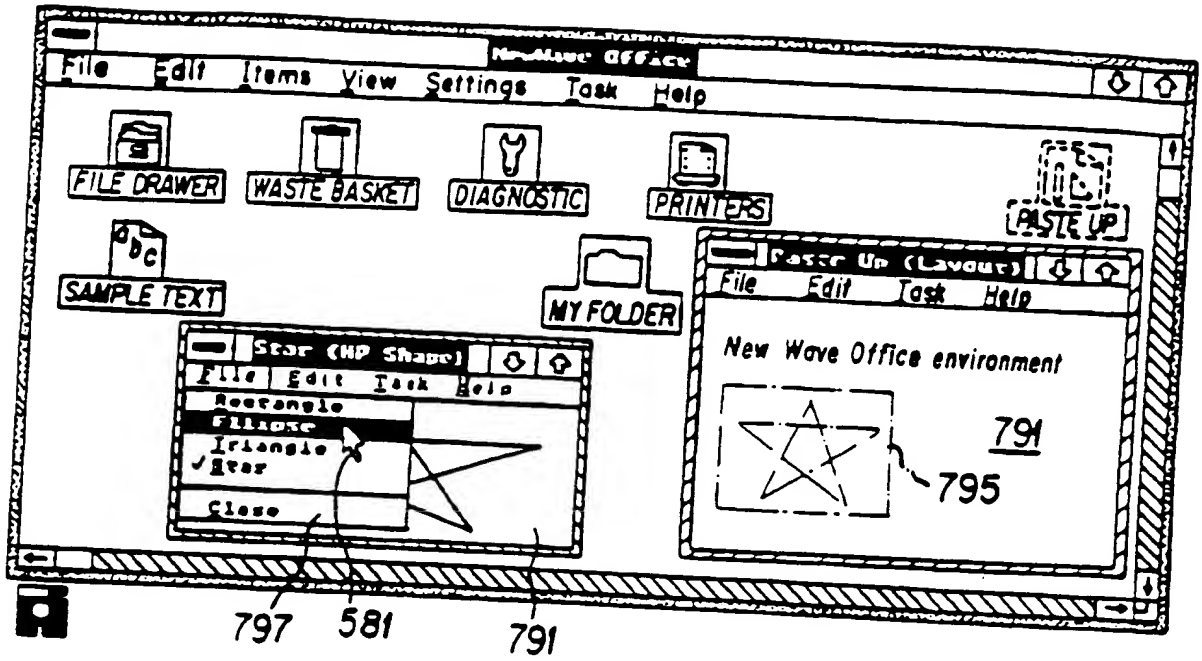

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Star (HP Shape)
File Edit Task Help
Rectangle
Filltone
Triangle
Star
Close
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
791
795
797
581
791

FIG. 50

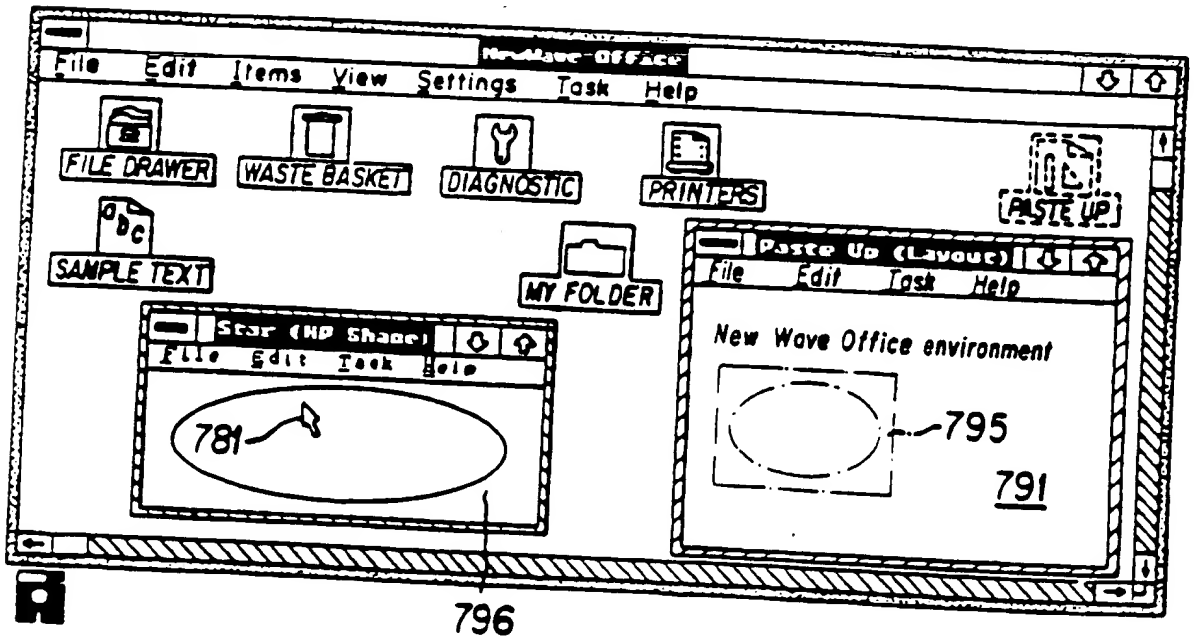

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Star (HP Shape)
File Edit Task Help
781
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
795
791
796

FIG. 51

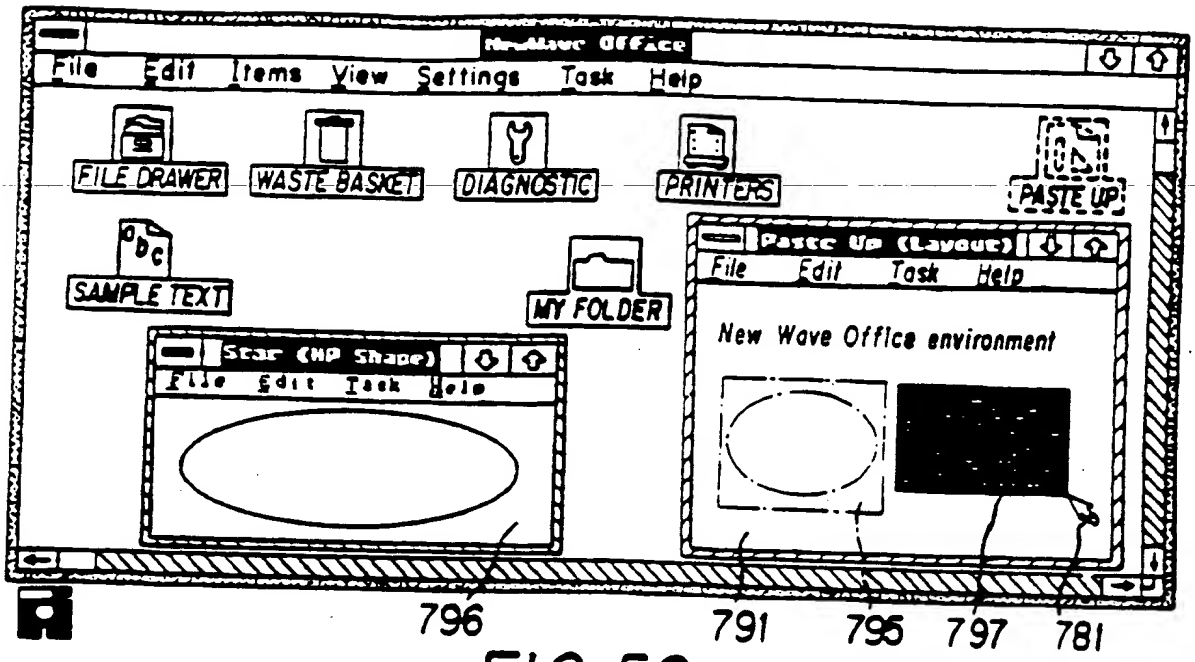

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
Star (HP Shape)
File Edit Task Help
796
791
795
797
781

FIG. 52

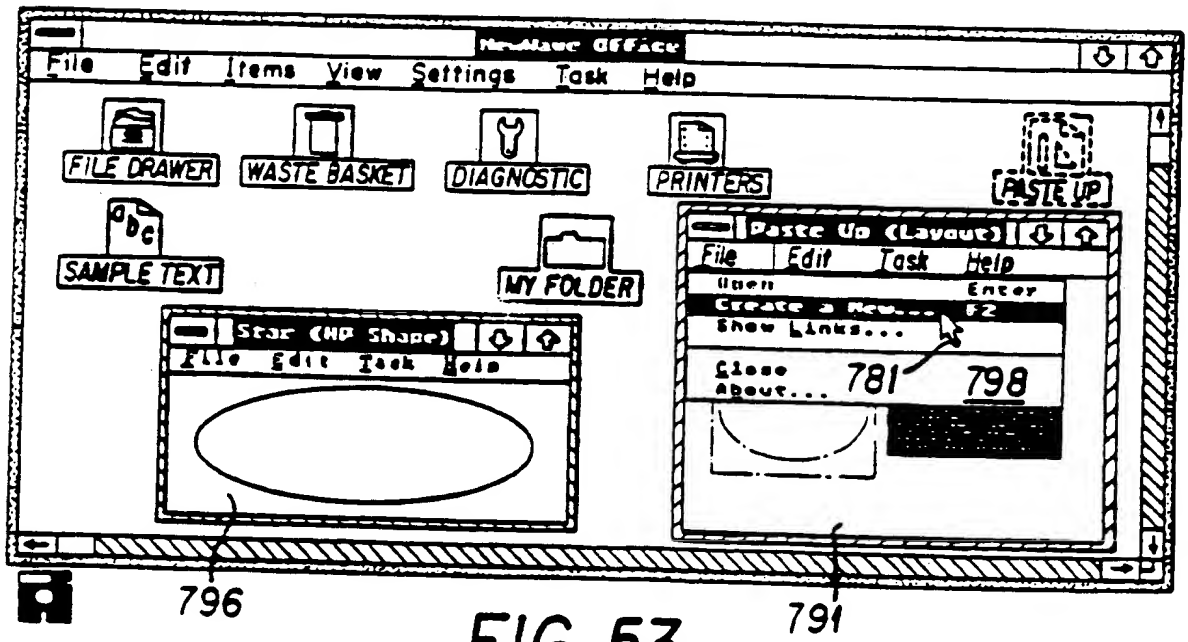

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
PASTE UP
SAMPLE TEXT
MY FOLDER
Paste Up (Layout)
File Edit Task Help
Create a New... F2
Show Links...
Close
About...
781
798
Star (HP Shape)
File Edit Task Help
796
791

FIG. 53

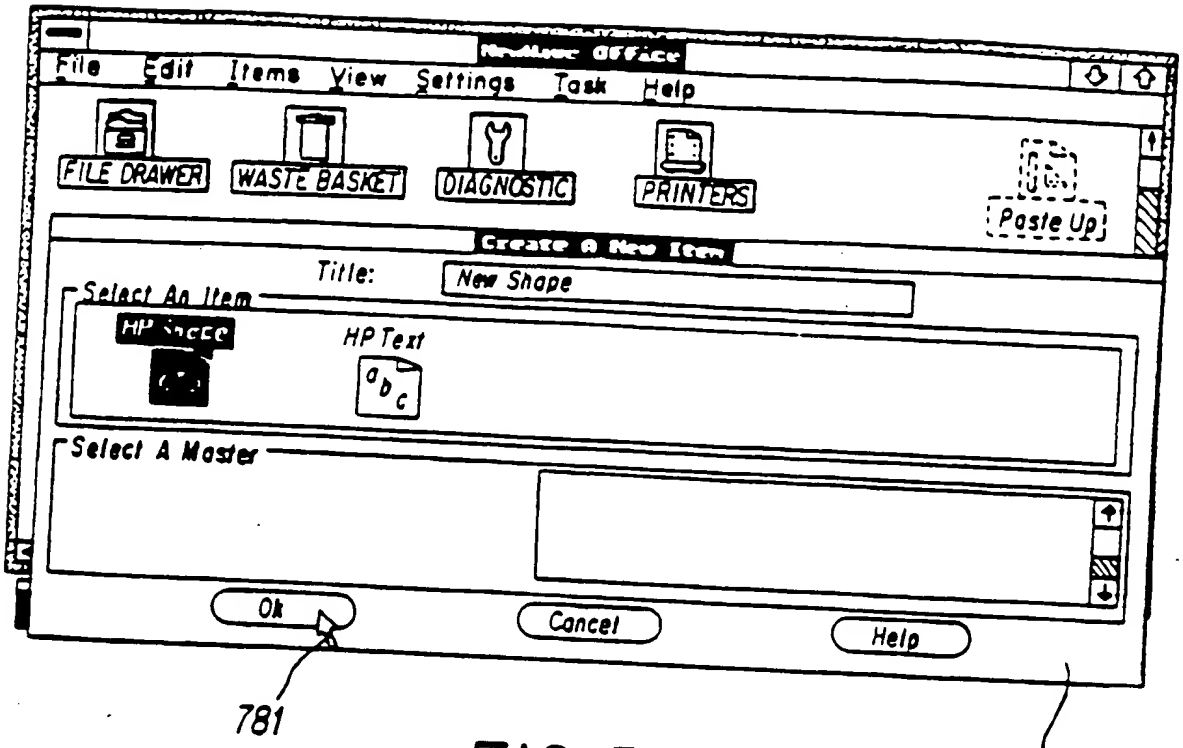

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Create A New Item
Title:
New Shape
Select An Item
HP Text
Select A Master
Ok
Cancel
Help
781
799

FIG. 54

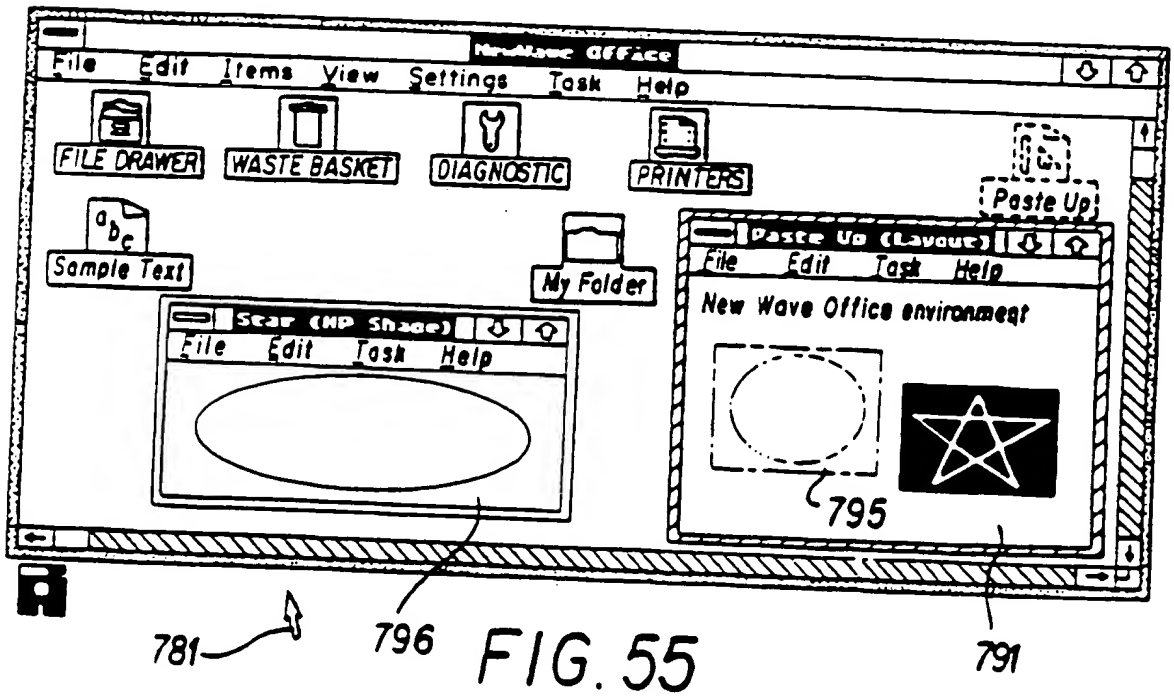

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
File Edit Task Help
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
795
781
796
791

FIG. 55

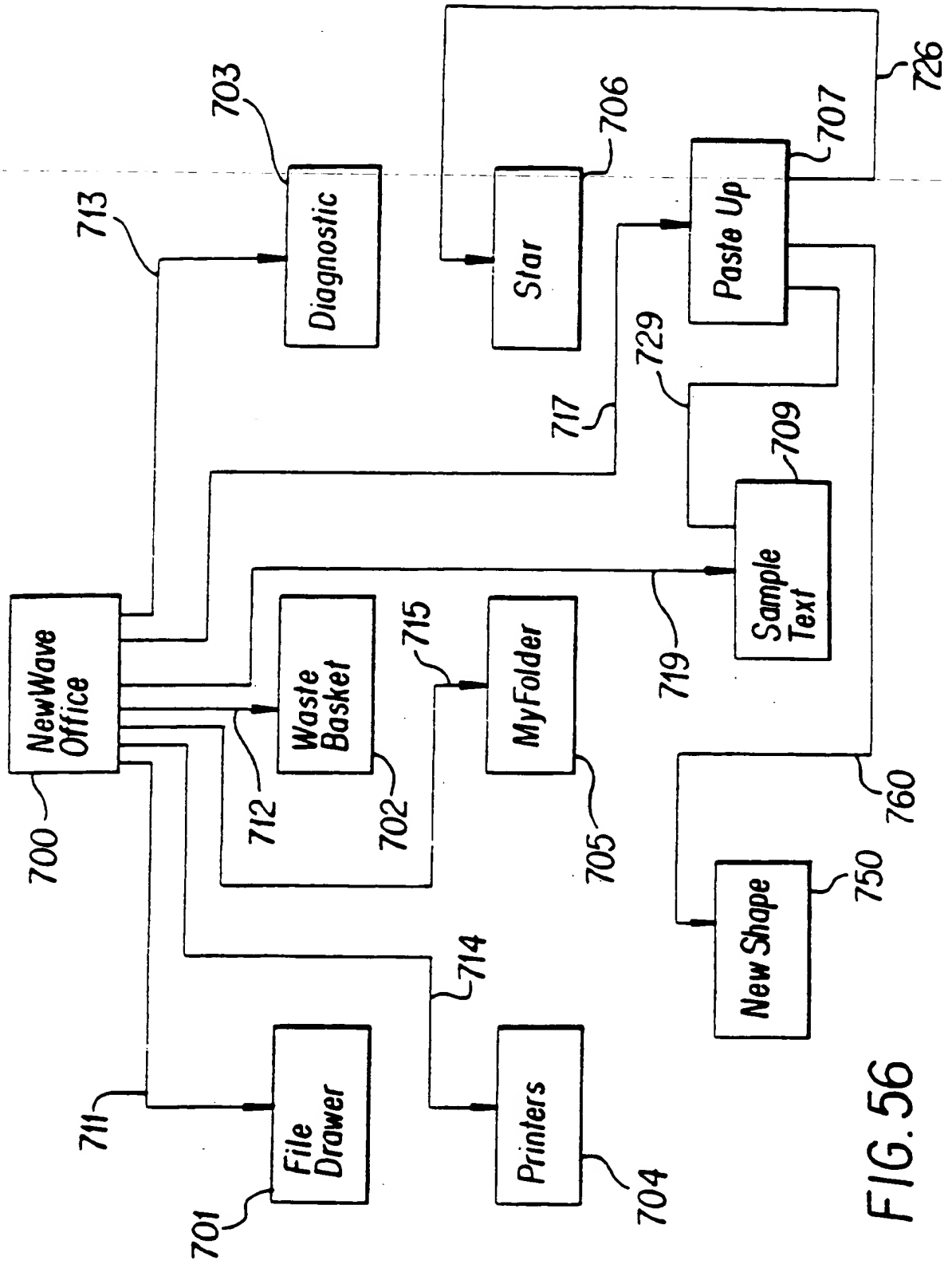

NewWave Office
Waste Basket
MyFolder
Sample Text
Paste Up
Star
Diagnostic
File Drawer
Printers
New Shape
700
701
702
703
704
705
706
707
709
711
712
713
714
715
717
719
726
729
750
760

FIG. 56

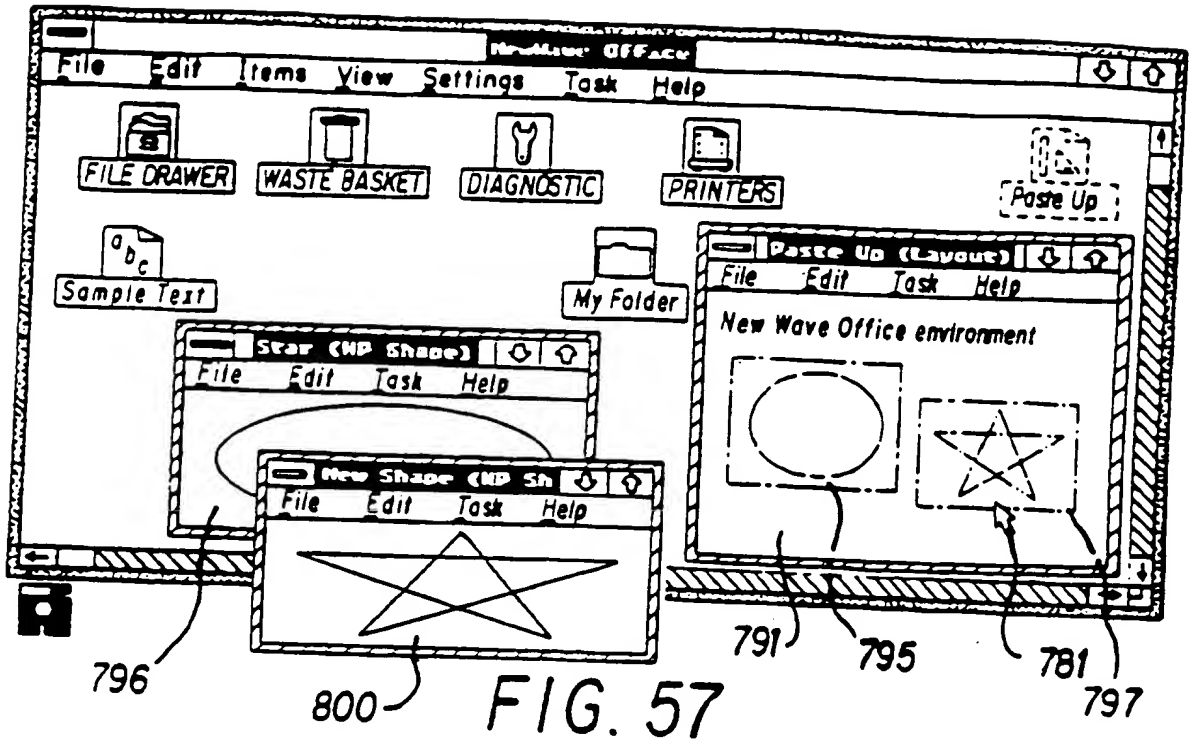

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Star (WP Shape)
File Edit Task Help
New Shape (WP Sh
File Edit Task Help
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
796
800
791
795
781
797

FIG. 57

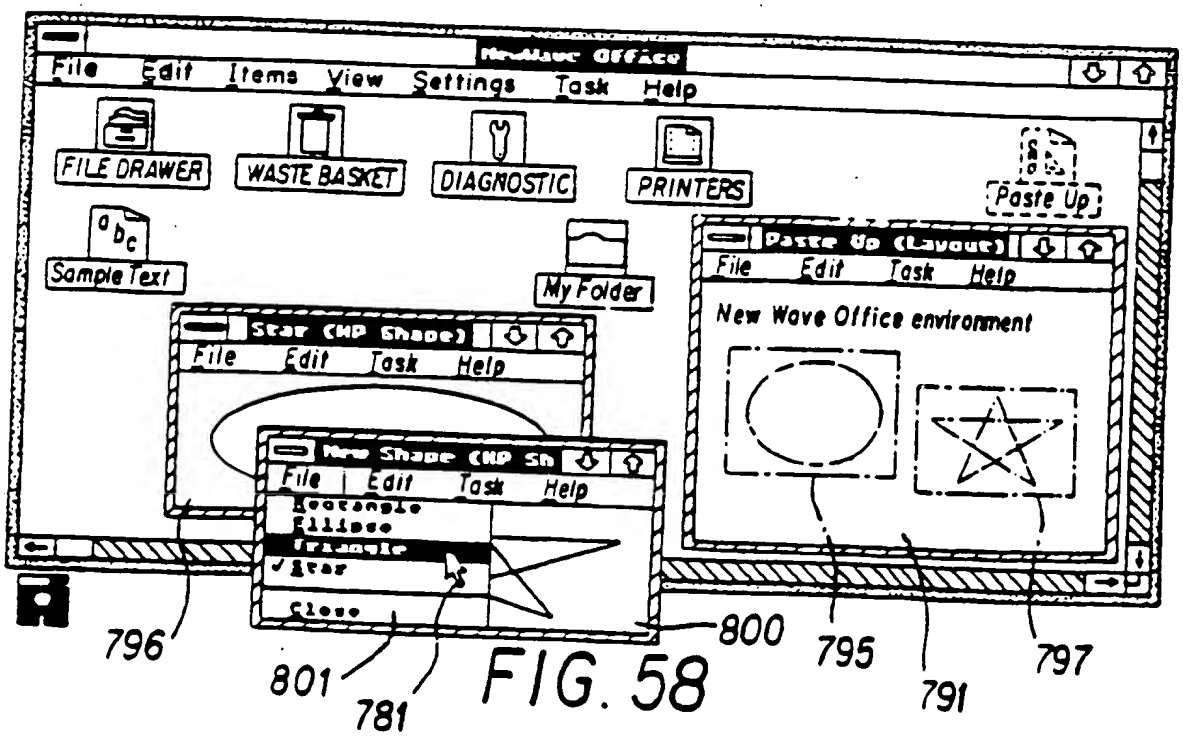

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Star (WP Shape)
File Edit Task Help
New Shape (WP Sh
File Edit Task Help
Rectangle
Ellipse
Triangle
Star
Close
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
796
801
781
800
795
791
797

FIG. 58

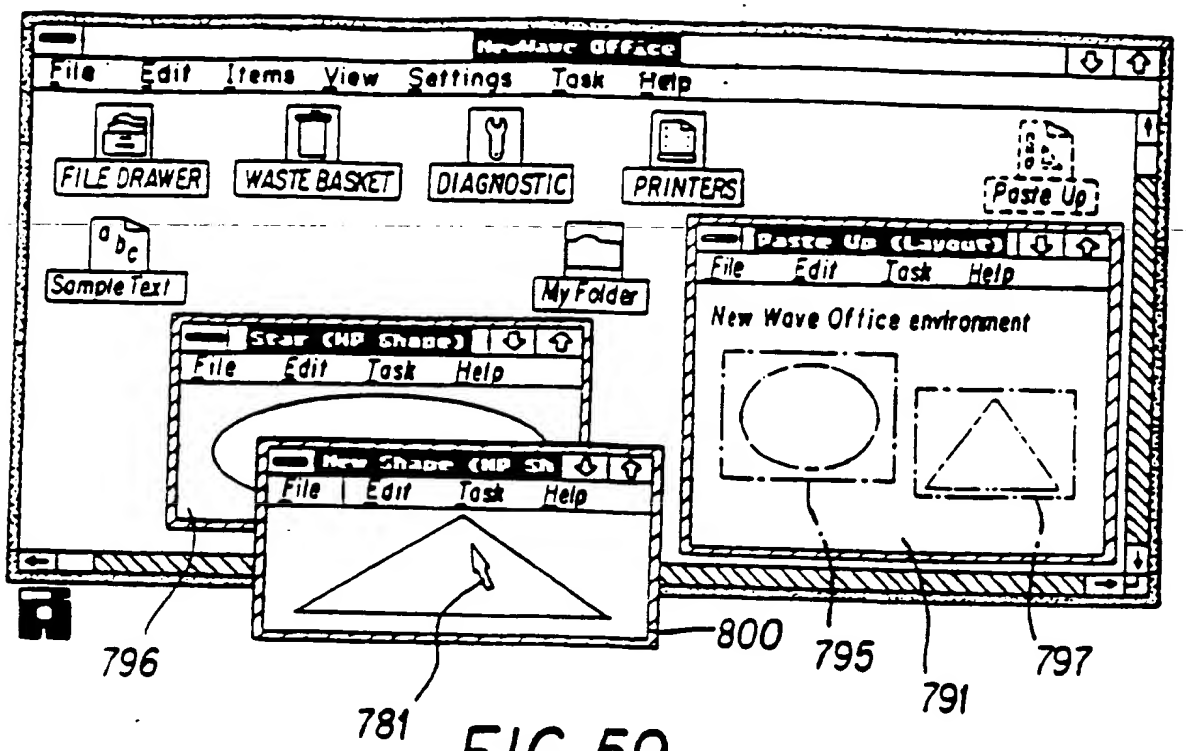

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
File Edit Task Help
New Wave Office environment
796
781
800
795
791
797

FIG. 59

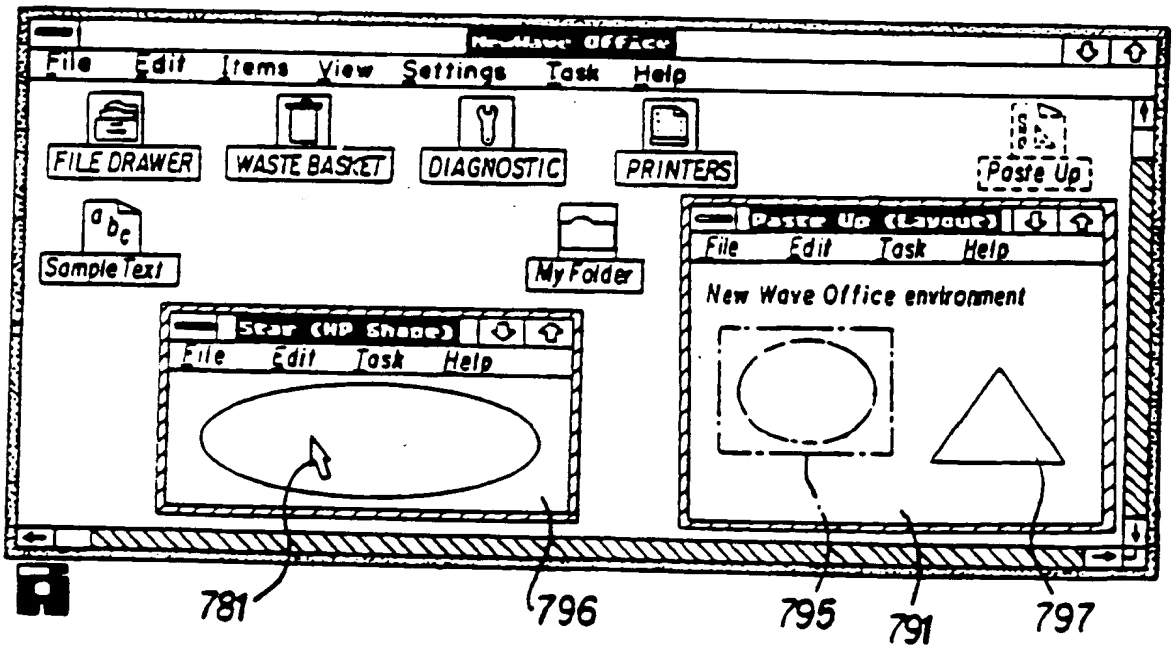

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
File Edit Task Help
New Wave Office environment
781
796
795
791
797

FIG. 60

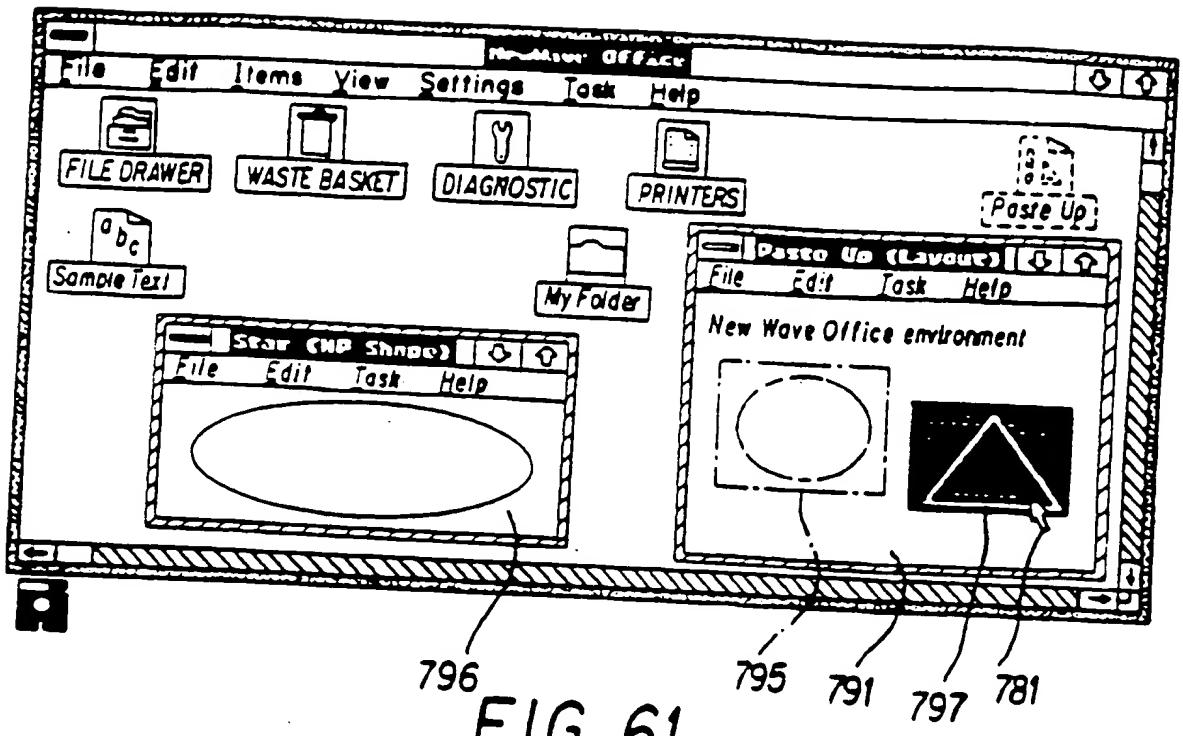

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
Star (WP Shape)
File Edit Task Help
796
795
791
797
781

FIG. 61

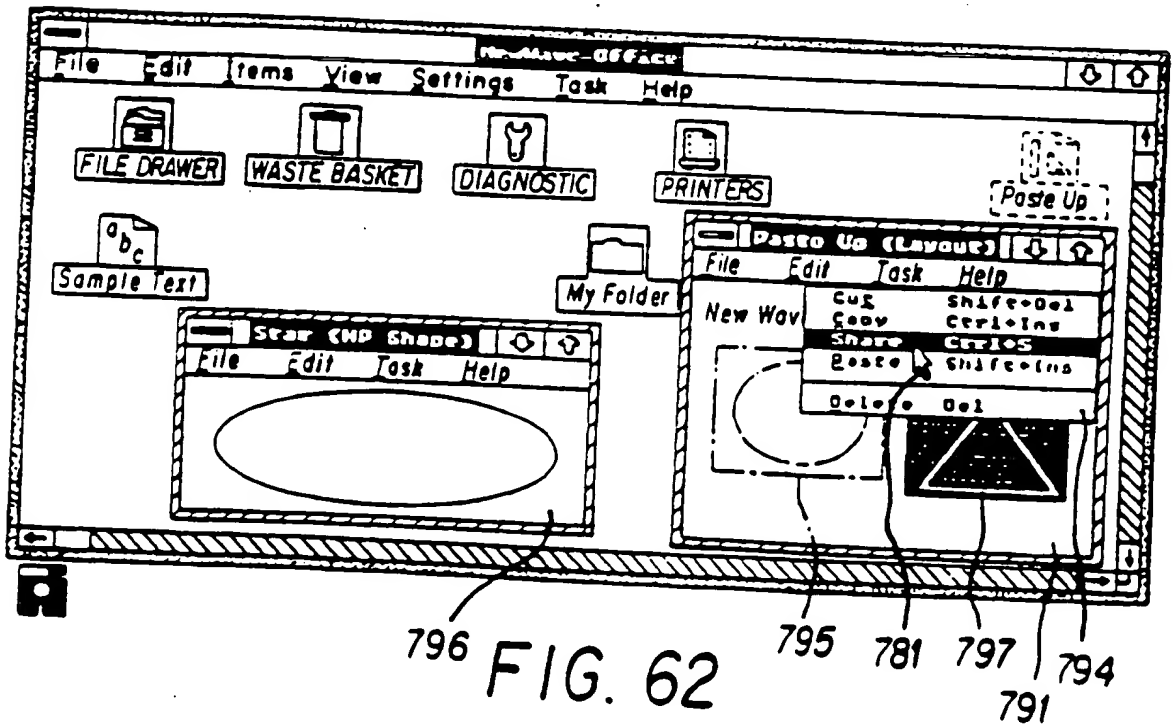

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
Cut Shift+Del
Copy Ctrl+Ins
Paste Shift+Ins
Delete Del
Star (WP Shape)
File Edit Task Help
796
795
781
797
794
791

FIG. 62

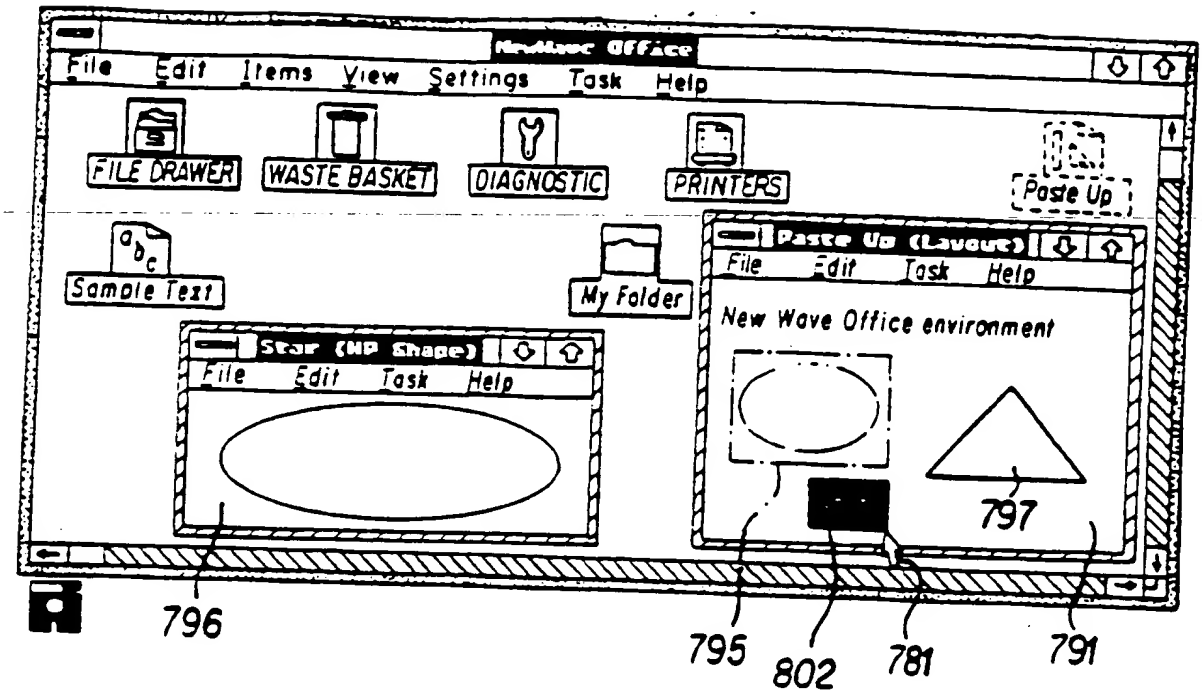

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
Star (HP Shape)
File Edit Task Help
797
796
795
802
781
791

FIG. 63

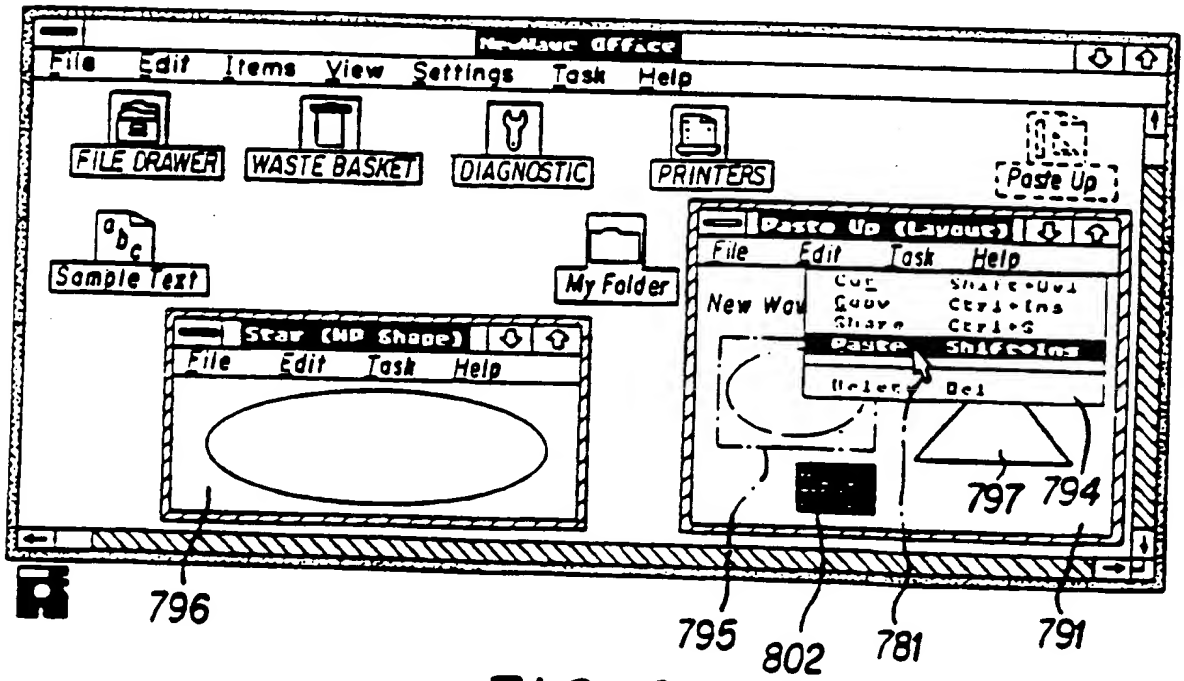

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
Star (HP Shape)
File Edit Task Help
797
794
796
795
802
781
791

FIG. 64

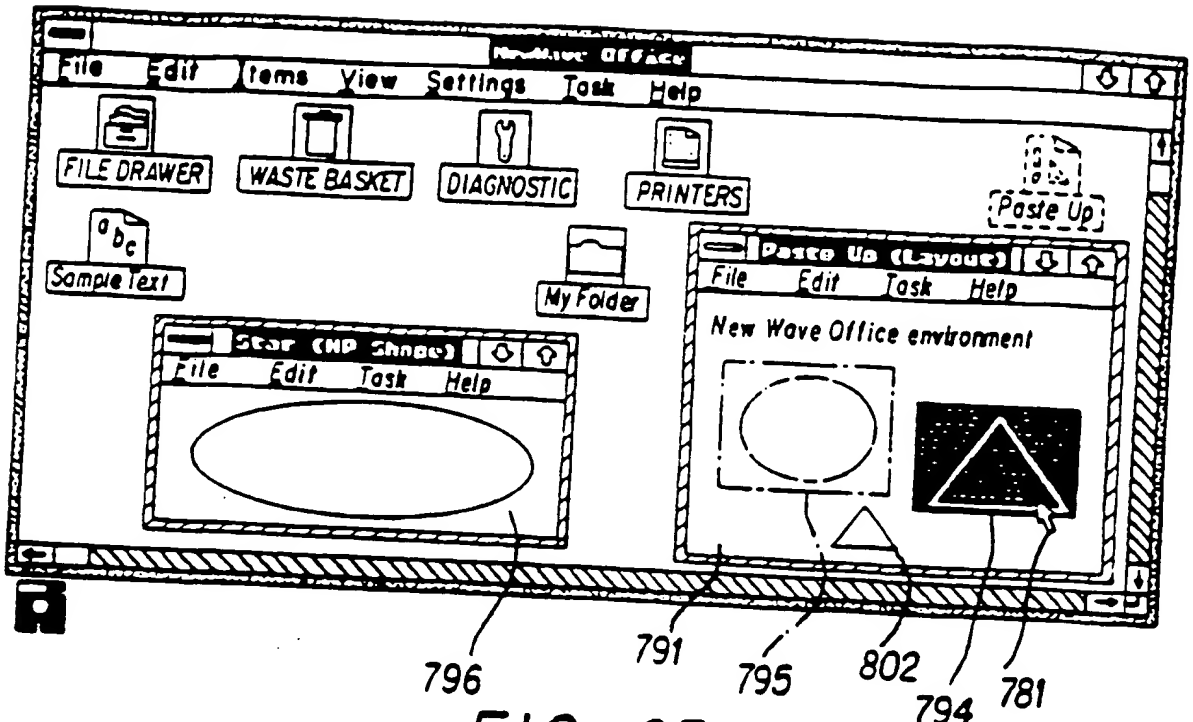

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
Star (NP Shapes)
File Edit Task Help
796
791
795
802
794
781

FIG. 65

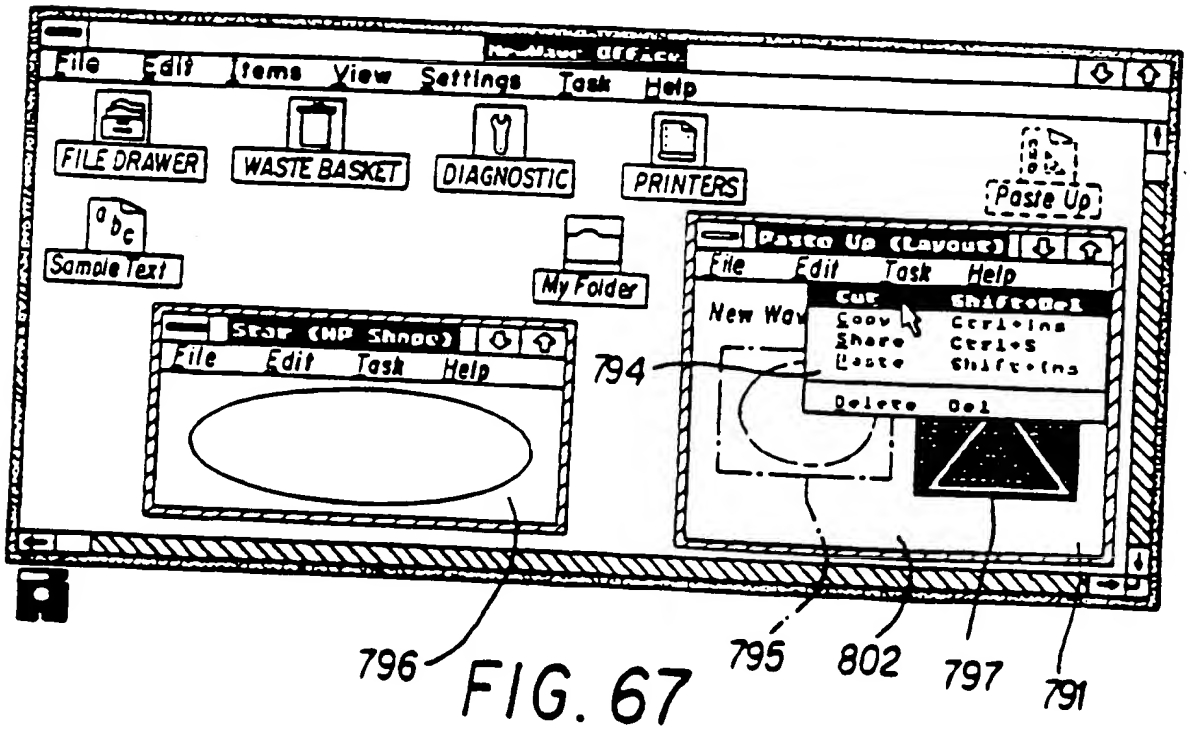

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
Cut
Delete Del
Star (NP Shapes)
File Edit Task Help
794
796
795
802
797
791

FIG. 67

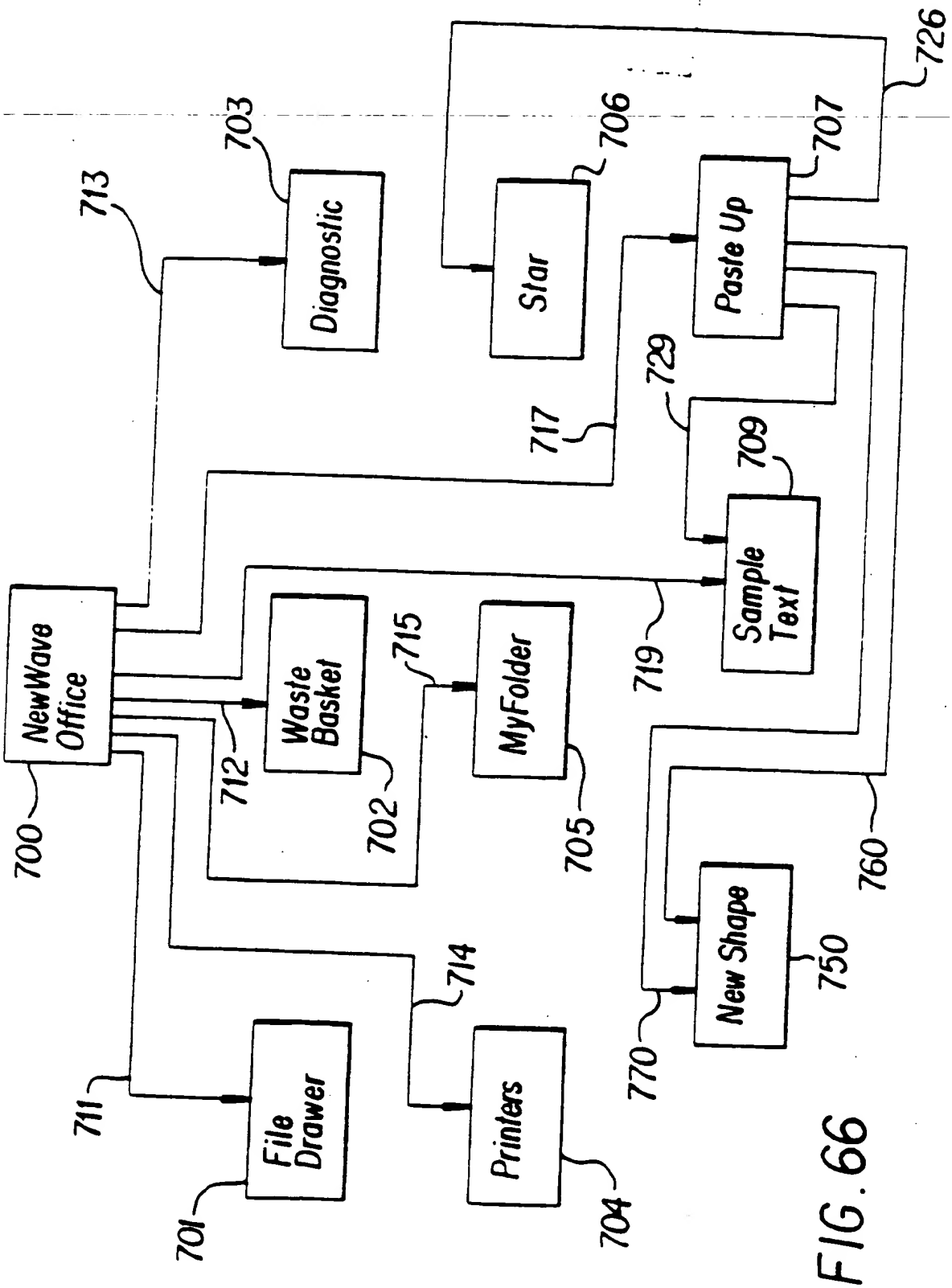

NewWave Office
700
Diagnostic
703
713
Star
706
726
Paste Up
707
729
717
709
Sample Text
719
Waste Basket
712
702
715
MyFolder
705
760
New Shape
750
770
714
File Drawer
711
701
Printers
704

FIG. 66

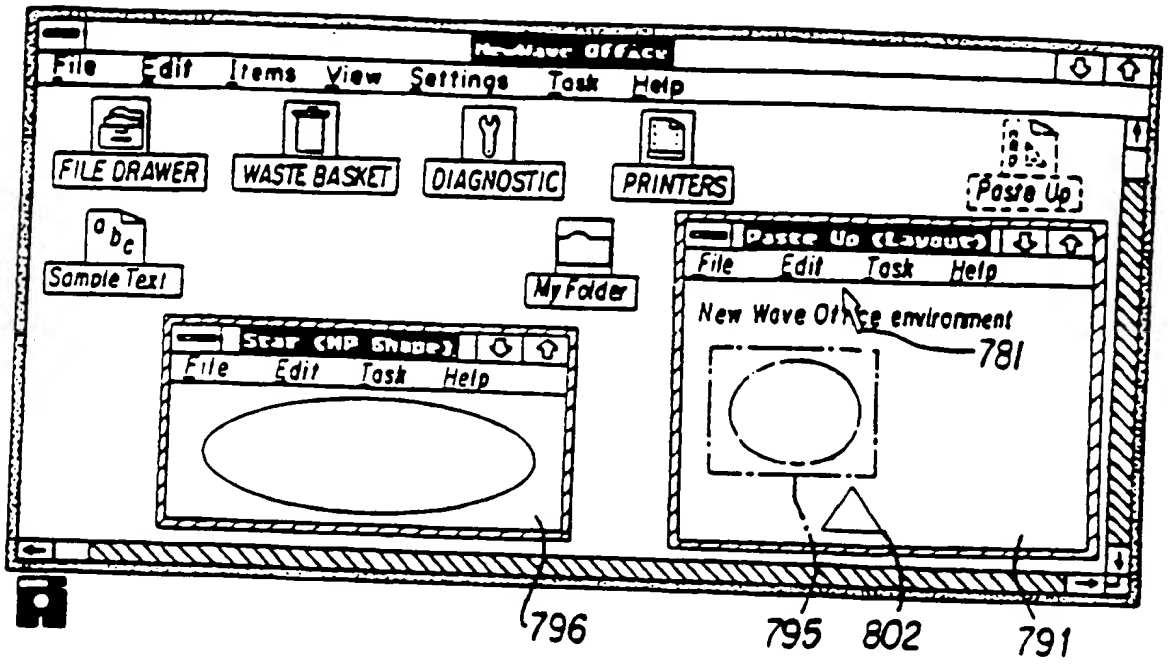

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
781
Star (NP Shape)
File Edit Task Help
796
795
802
791

FIG. 68

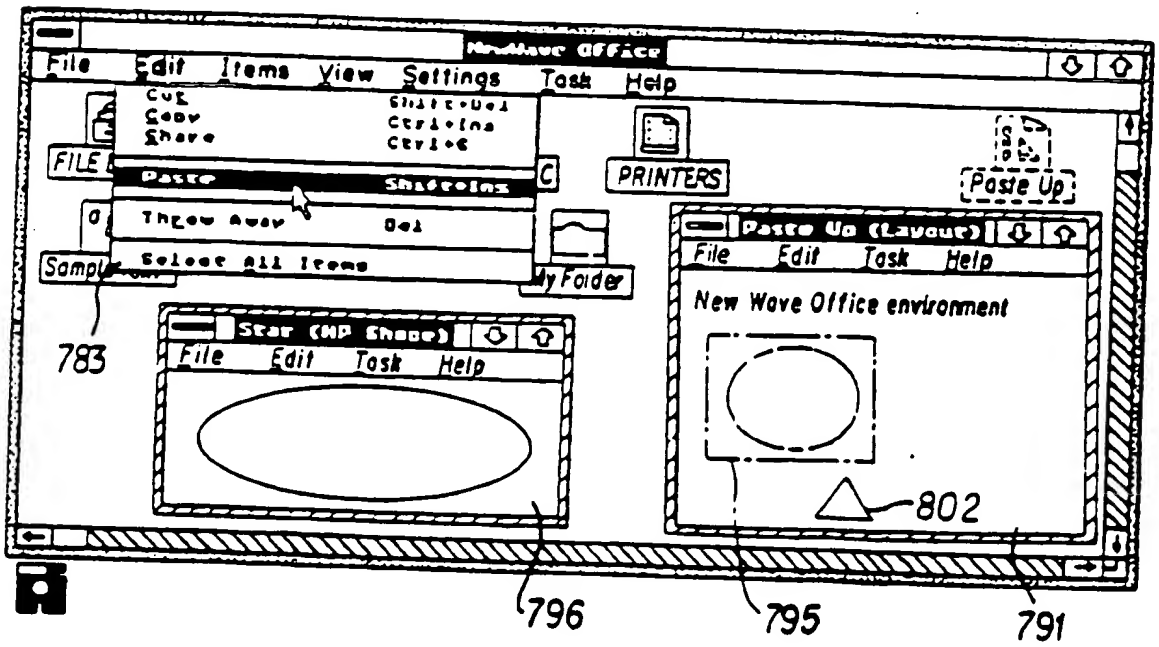

File Edit Items View Settings Task Help
Cut
Copy
Paste
Throw Away
Select All Items
PRINTERS
Paste Up
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
783
Star (NP Shape)
File Edit Task Help
802
796
795
791

FIG. 69

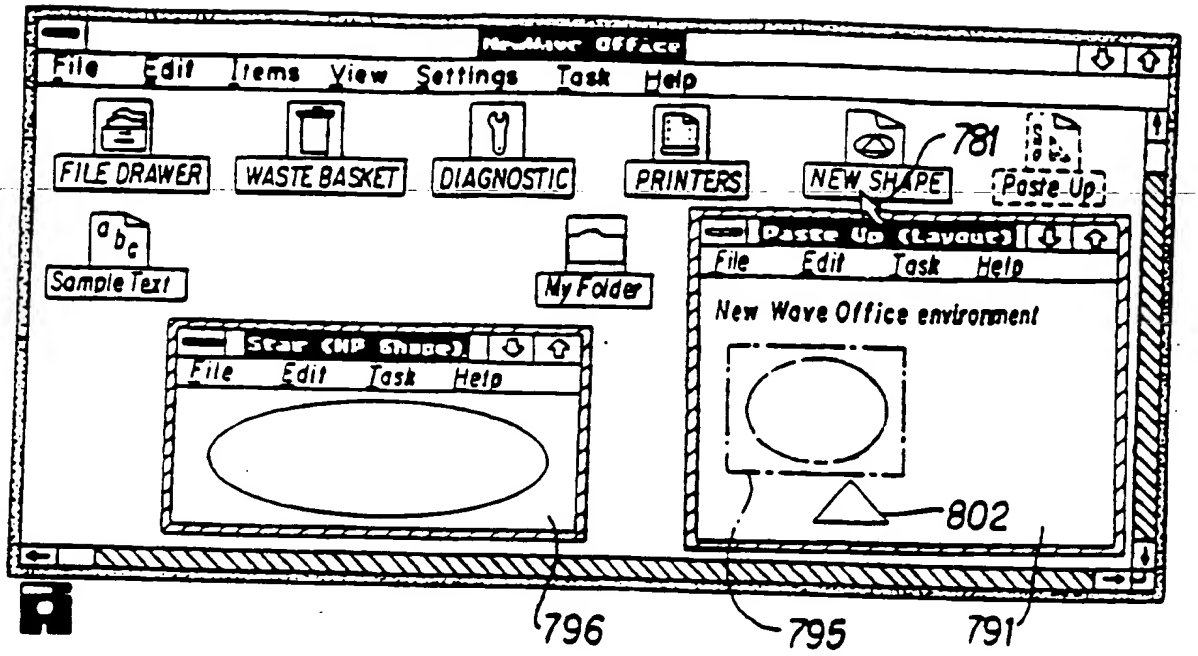

File Edit Items View Settings Task Help
FILE DRAWER
WASTE BASKET
DIAGNOSTIC
PRINTERS
NEW SHAPE
781
Paste Up
Sample Text
My Folder
Paste Up (Layout)
File Edit Task Help
New Wave Office environment
802
Star (NP Shape)
File Edit Task Help
796
795
791

FIG. 70

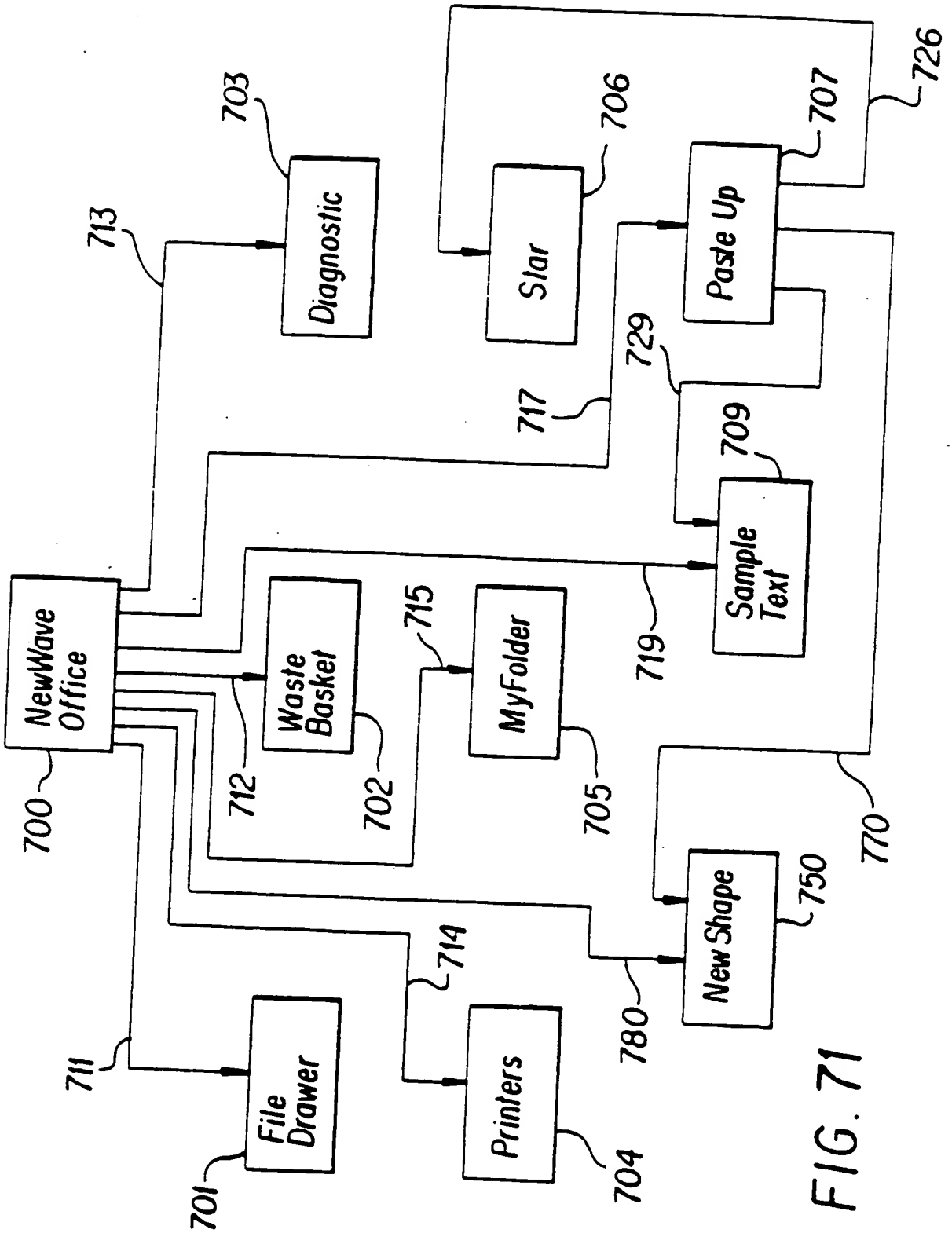

NewWave Office
700
Waste Basket
712
MyFolder
715
702
705
Sample Text
719
709
Paste Up
707
729
717
Star
706
Diagnostic
703
713
726
770
New Shape
750
780
File Drawer
711
701
Printers
714
704

FIG. 71

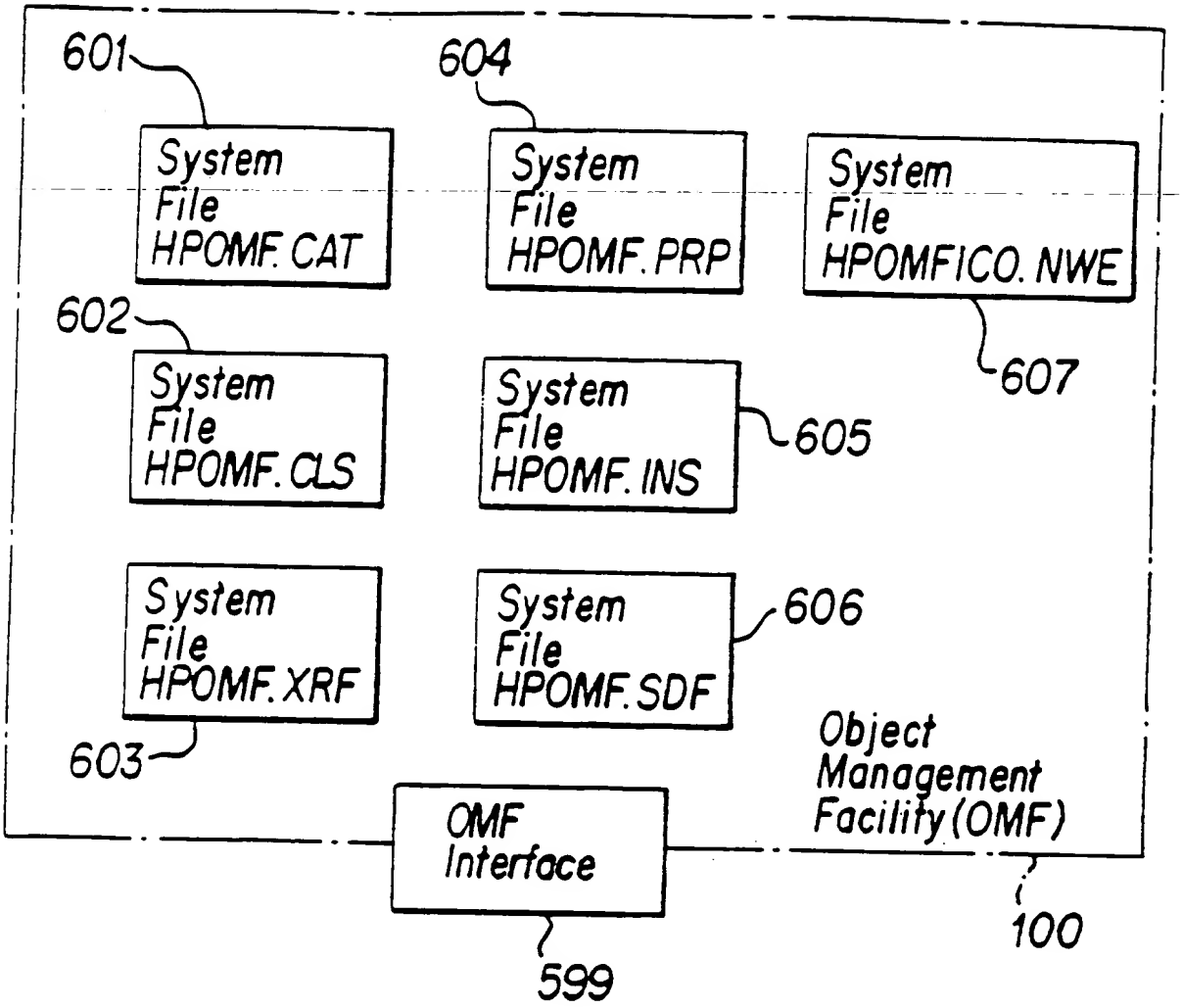

601
System File HPOMF.CAT
604
System File HPOMF.PRP
System File HPOMFICO.NWE
607
602
System File HPOMF.CLS
System File HPOMF.INS
605
System File HPOMF.XRF
603
System File HPOMF.SDF
606
Object Management Facility (OMF)
OMF Interface
599
100

FIG. 72

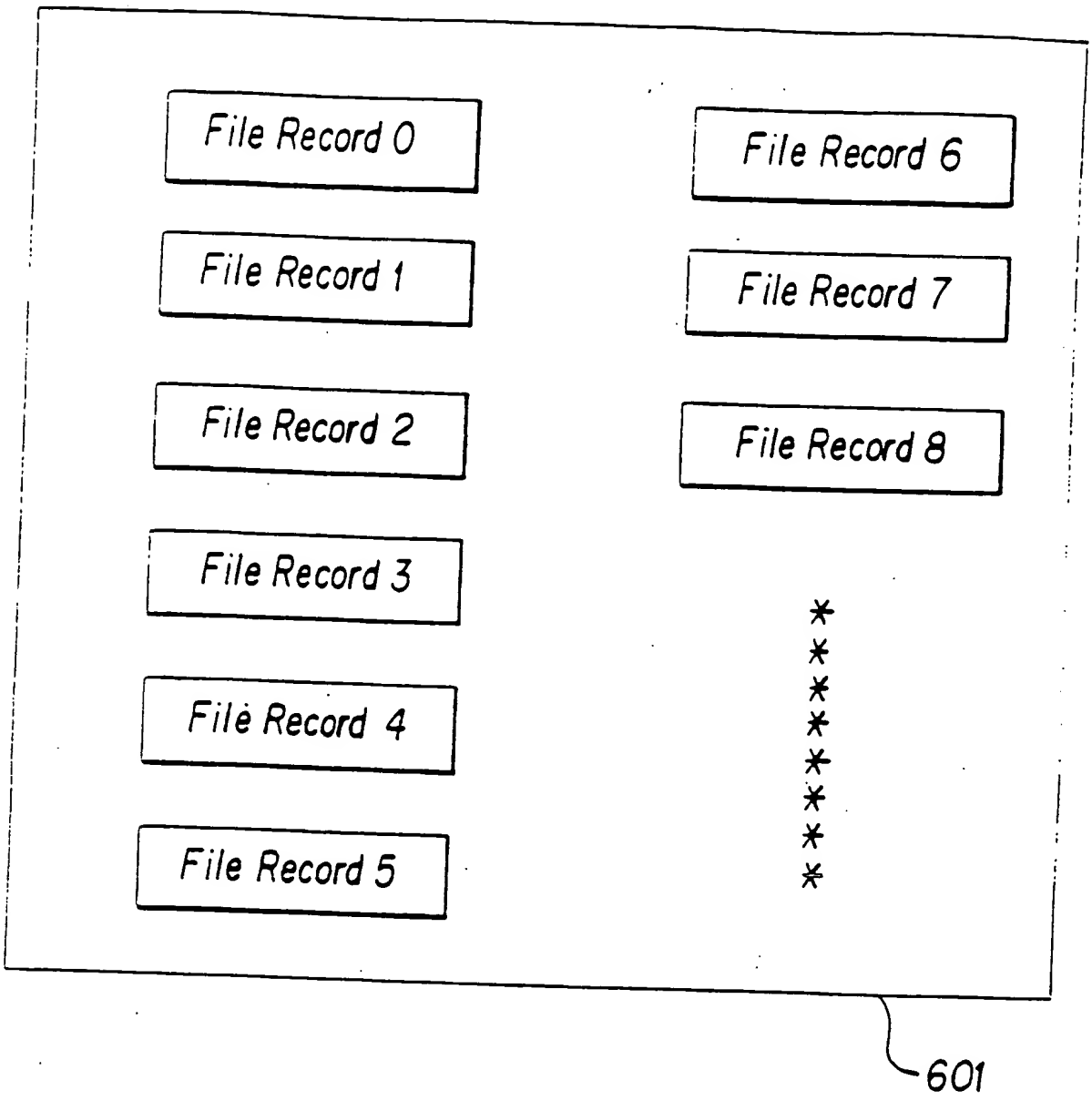

File Record 0
File Record 1
File Record 2
File Record 3
File Record 4
File Record 5
File Record 6
File Record 7
File Record 8
601

FIG. 73

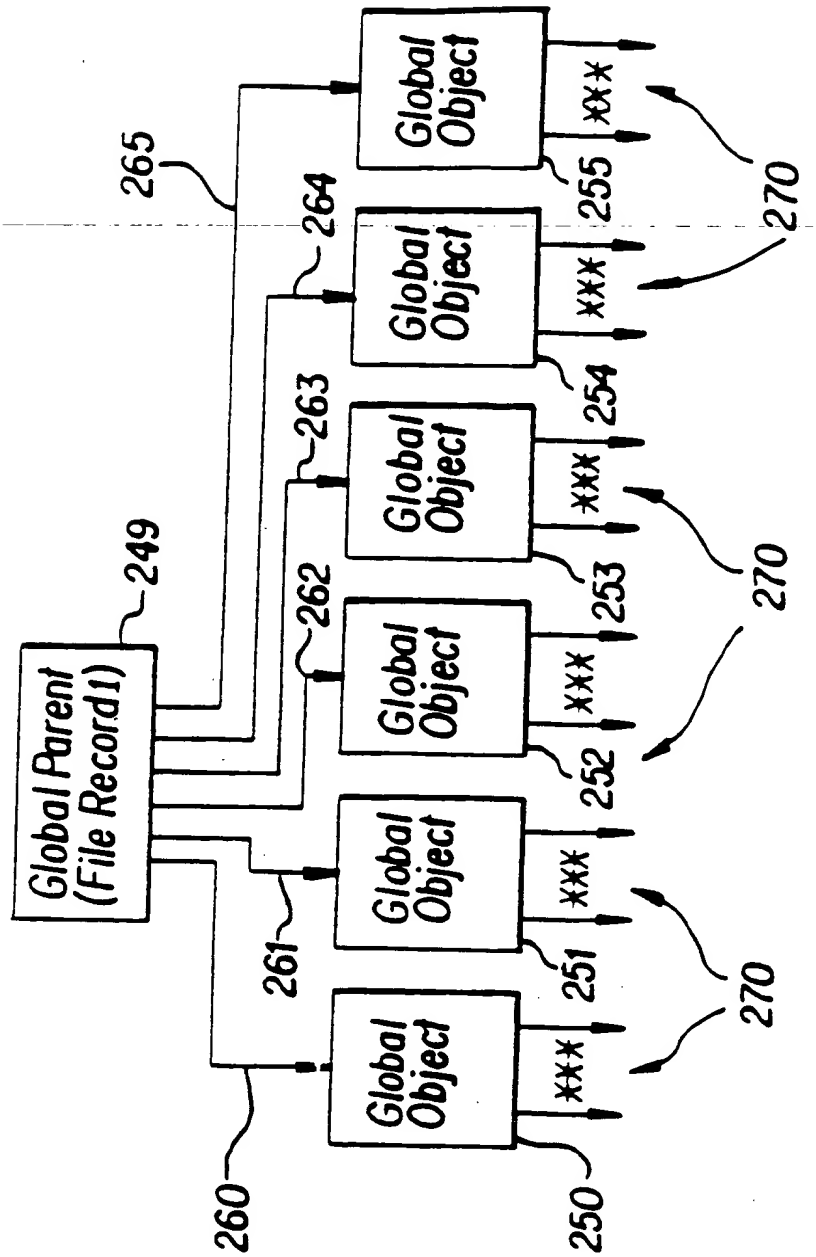

Global Parent (File Record1)
249
260
261
262
263
264
265
Global Object
***
250
251
252
253
254
255
270

FIG. 74

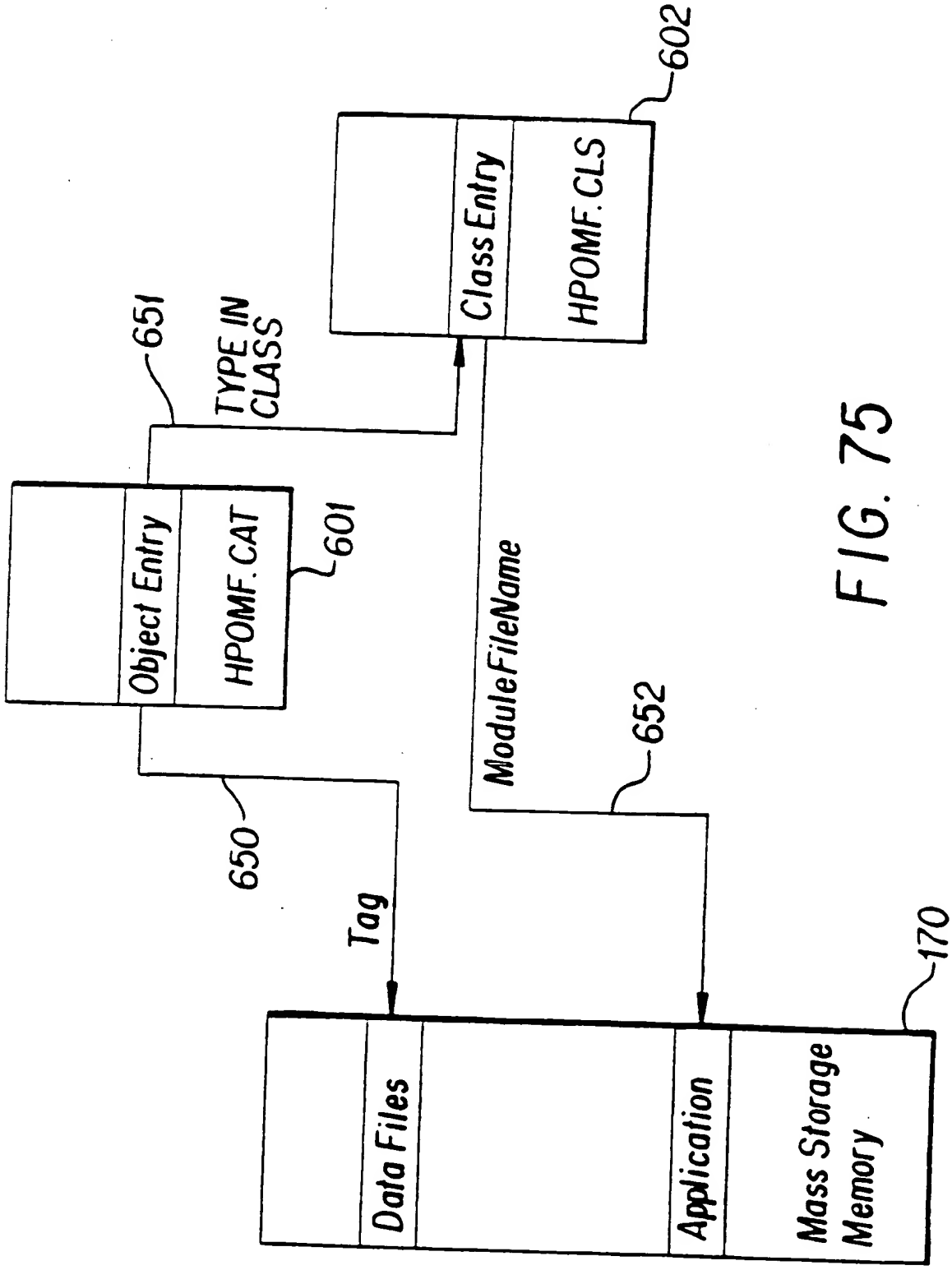

602
Class Entry
HPOMF.CLS
651
TYPE IN CLASS
Object Entry
HPOMF.CAT
601
ModuleFileName
652
650
Tag
170
Data Files
Application
Mass Storage Memory

FIG. 75

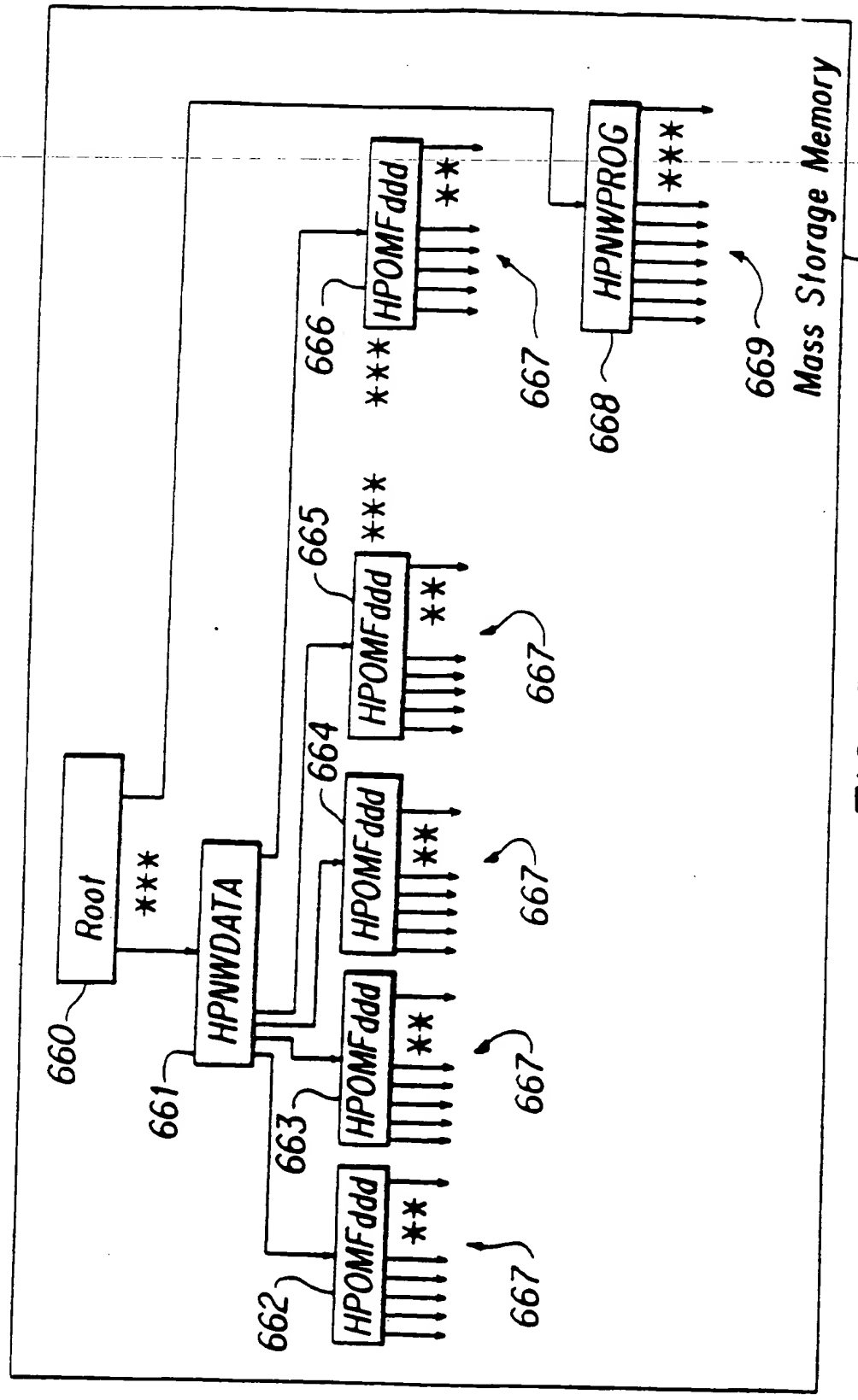

Root
***
660
HPNWDATA
661
HPOMFddd
662
HPOMFddd
663
HPOMFddd
664
HPOMFddd
665
HPOMFddd
666
***
667
HPNWPROG
668
669
Mass Storage Memory
170

FIG. 76

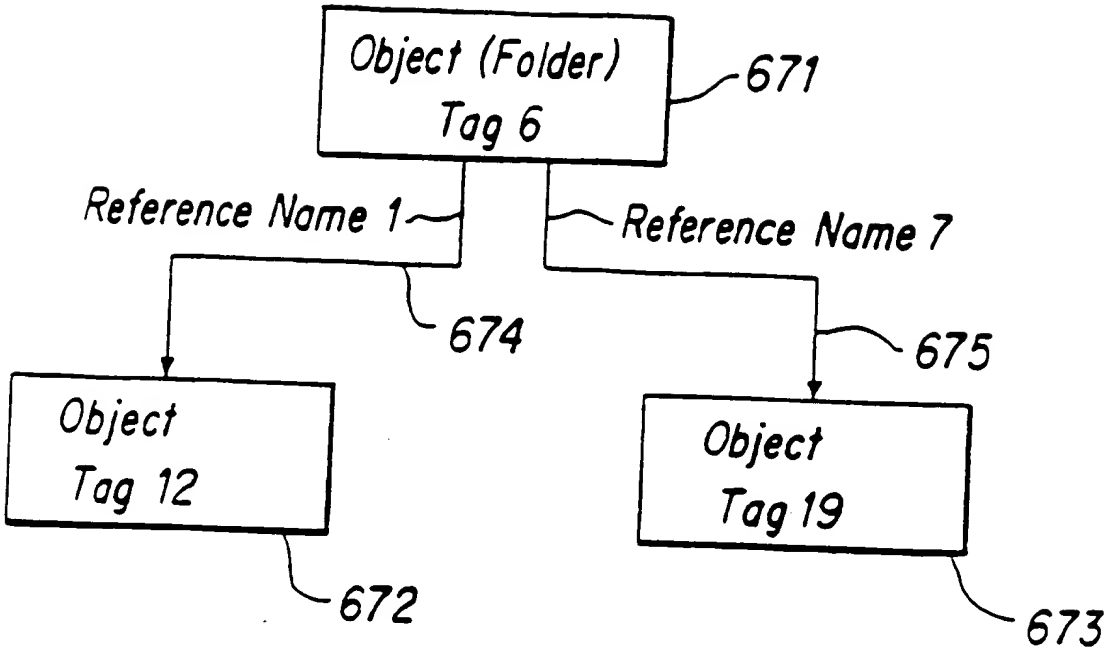

Object (Folder)
Tag 6
671
Reference Name 1
Reference Name 7
674
675
Object
Tag 12
Object
Tag 19
672
673

FIG. 77

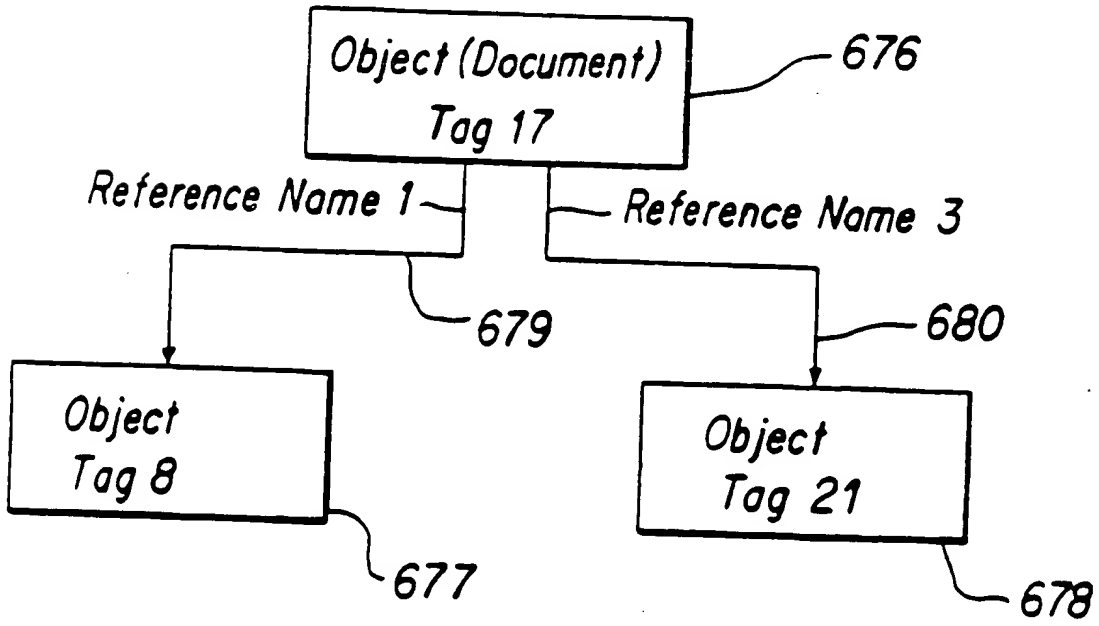

Object (Document)
Tag 17
676
Reference Name 1
Reference Name 3
679
680
Object
Tag 8
Object
Tag 21
677
678

FIG. 78

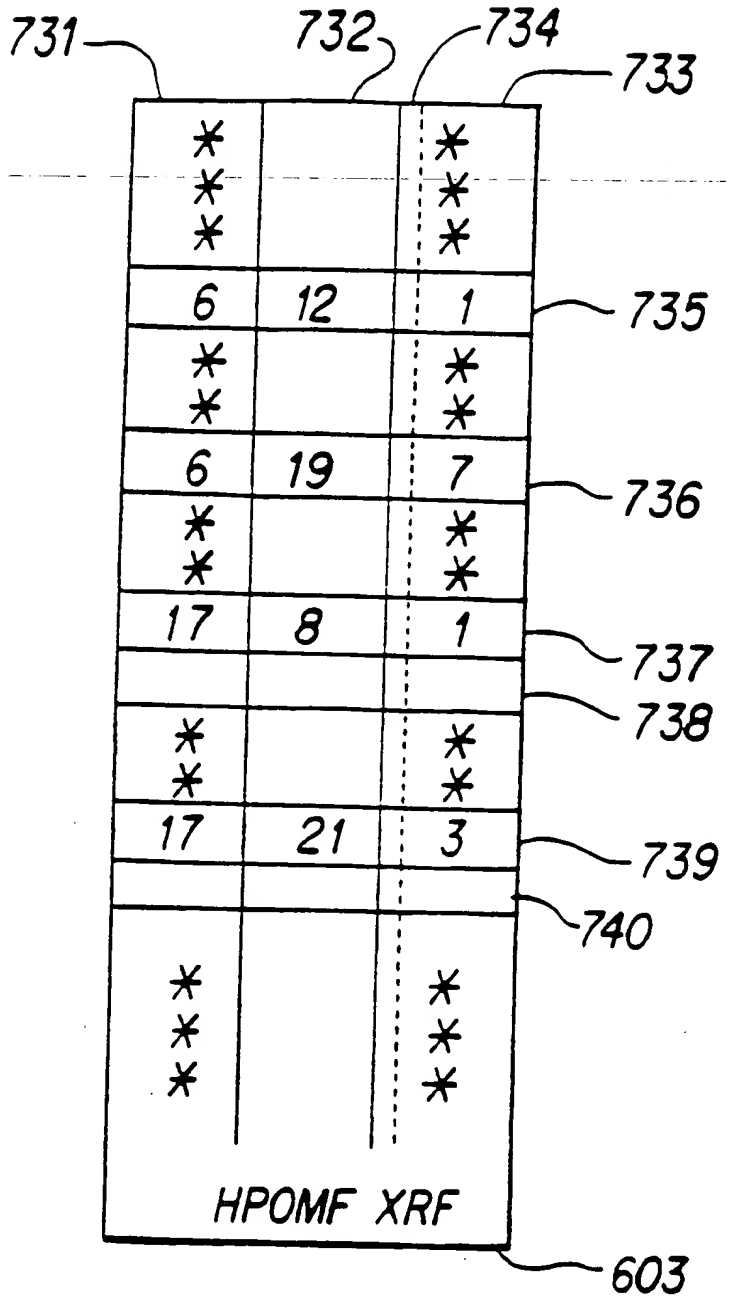
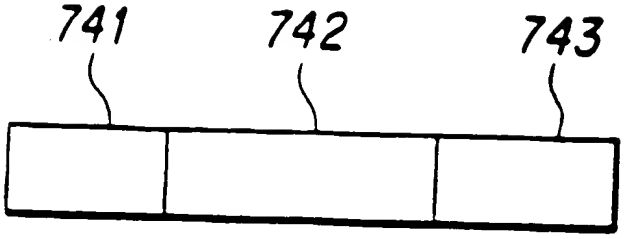

731
732
734
733
6
12
1
735
6
19
7
736
17
8
1
737
738
17
21
3
739
740
HPOMF XRF
603

FIG. 79



View Specification Record 740

FIG. 80

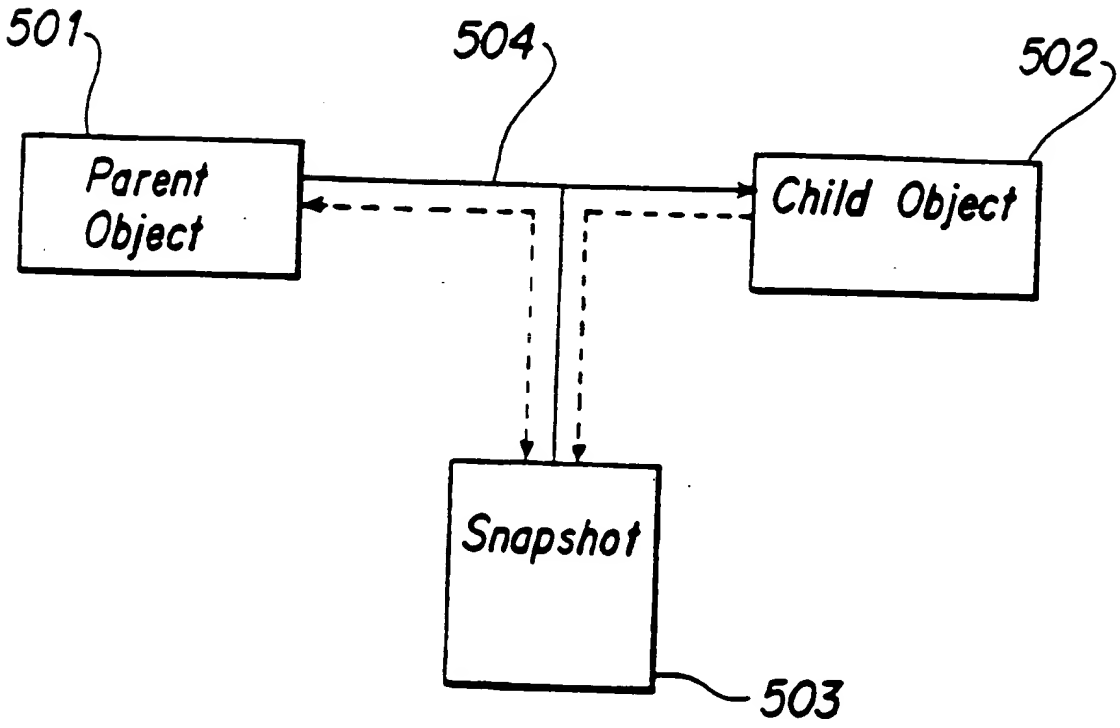


FIG. 81

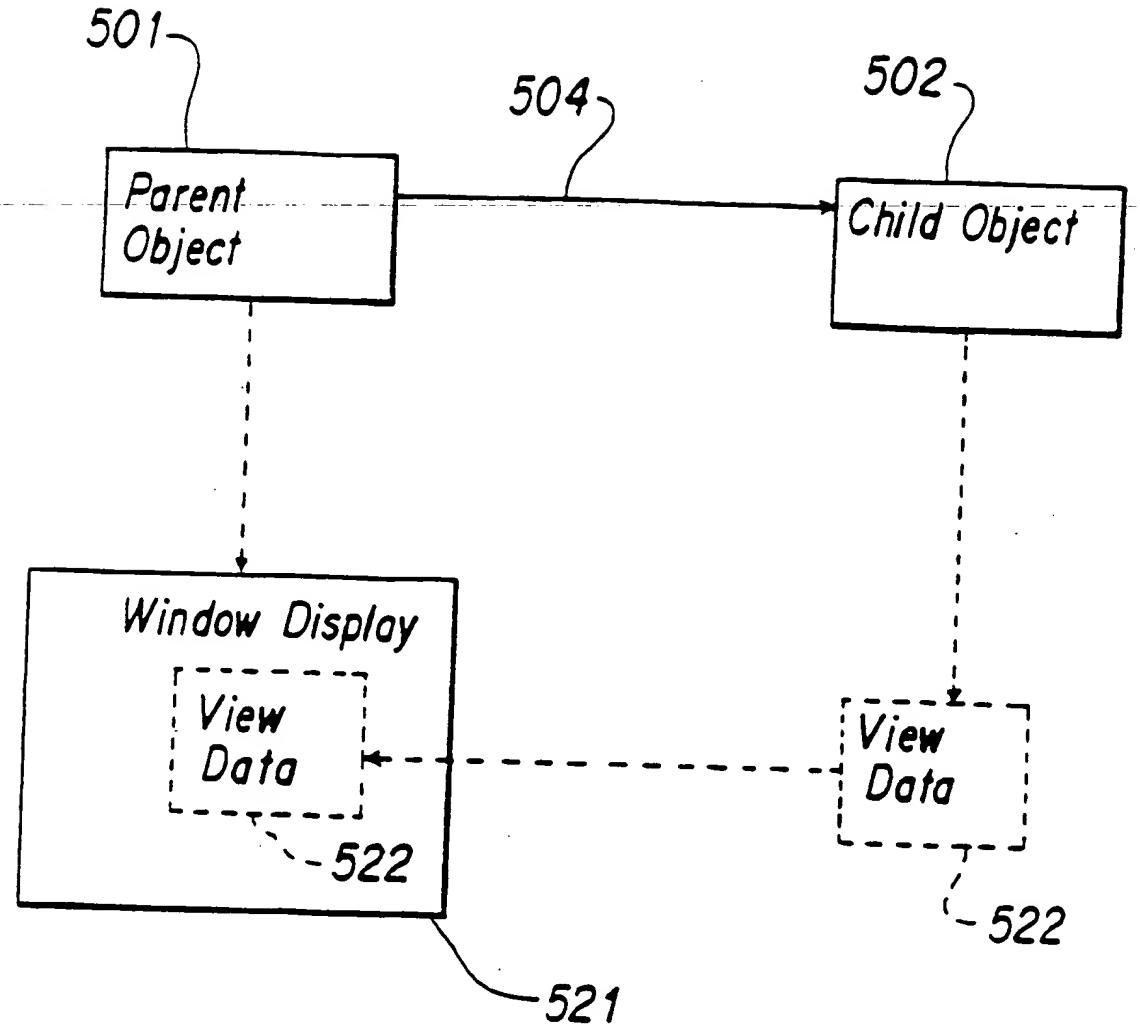

501
504
502
Parent Object
Child Object
Window Display
View Data
522
View Data
522
521

FIG. 82

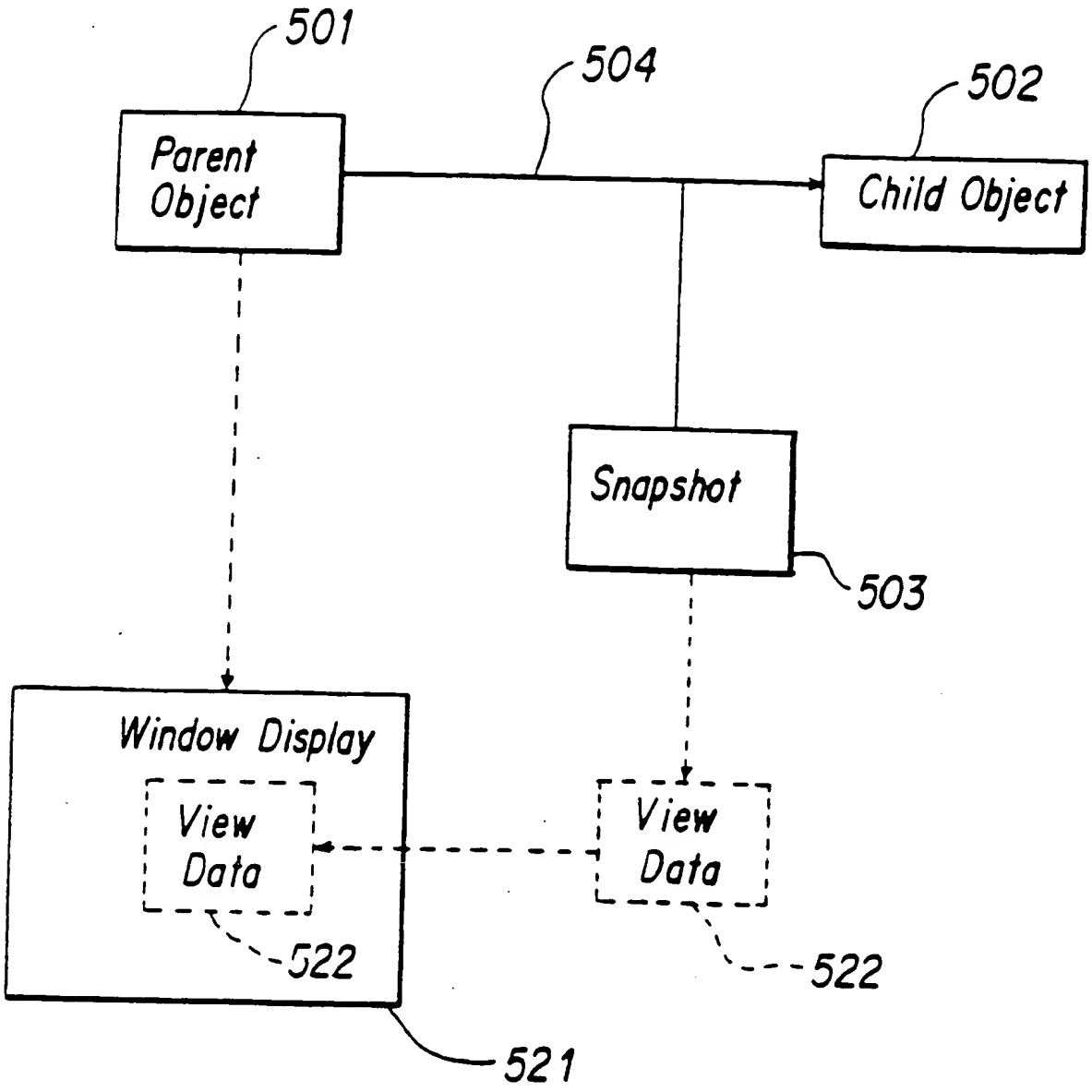


FIG. 83

Claims

1. An object based distributed computer system comprising a network of workstations and means for transmitting objects between workstations characterised by objects including a first object type for storing data and a second object type for presenting data to a user, wherein objects of the second type (V-c) reference an associated object of the first type (V-s) to enable a plurality of users of workstations to access data of the object of the first type, comprising means for transmitting an object of the second type (V-c) between workstations thereby to create a reference to the associated object of the first type (V-s) for each workstation receiving an object of the second type.

2. A system according to claim 1 comprising means for copying an object of the second type (V-c) between workstations.

3. A system according to claim 1 or claim 2 wherein transmitted objects of the second type (V-c)include an identifier (60) for the associated object of the first type (V-s).

4. A system according to any preceding claim in the form of a conferencing system comprising means enabling users of the workstations to participate in a meeting over the network wherein objects of the first type (V-s) store meeting data and objects of the second type (V-c) are for presenting meeting data.

5. A method of convening a meeting using a system as claimed in claim 4 comprising transmitting an object of the second type (V-c) between workstations thereby to create a reference to the associated object of the first type (V-s) for each workstation receiving an object of the second type.

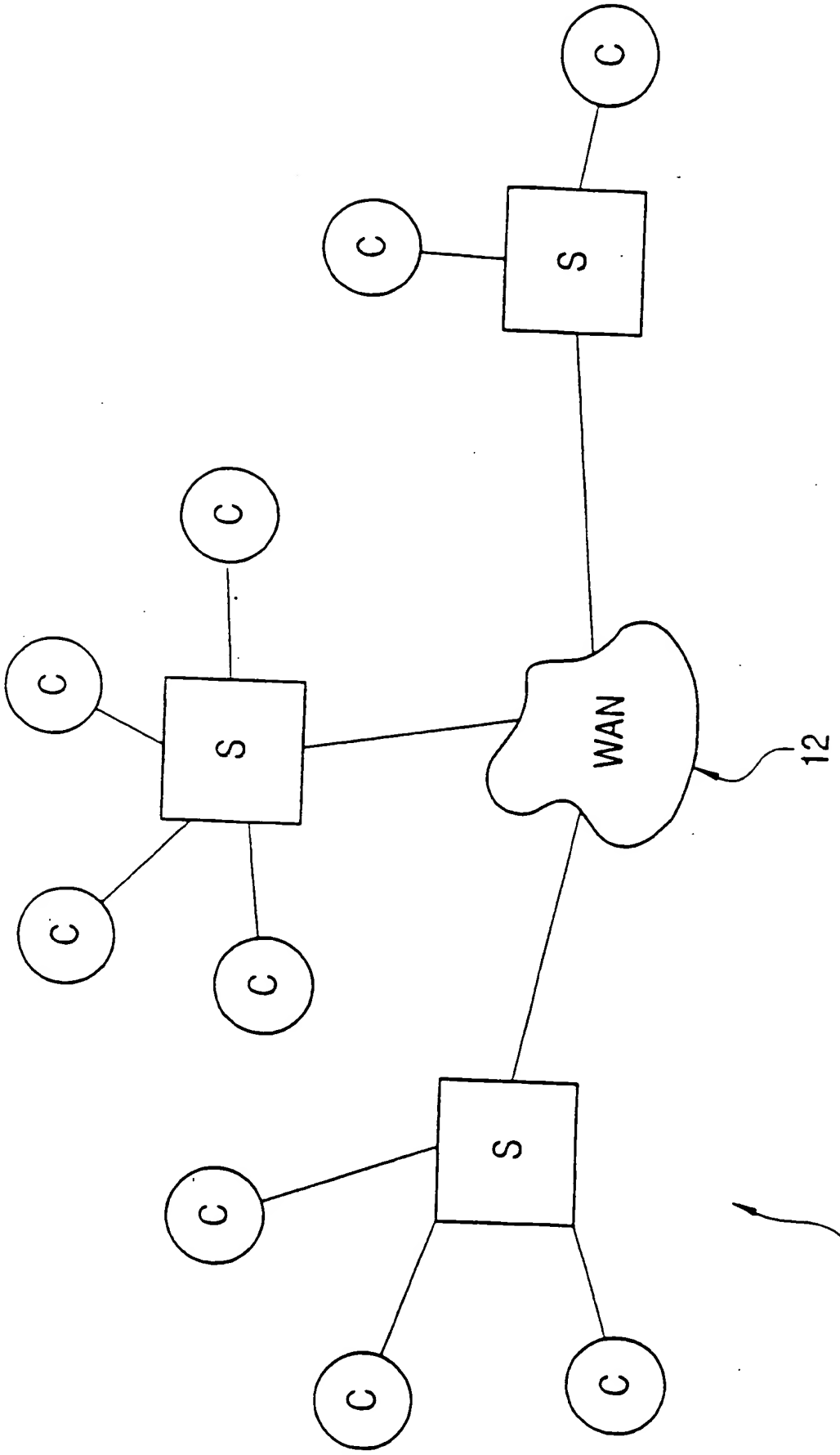


FIG 1

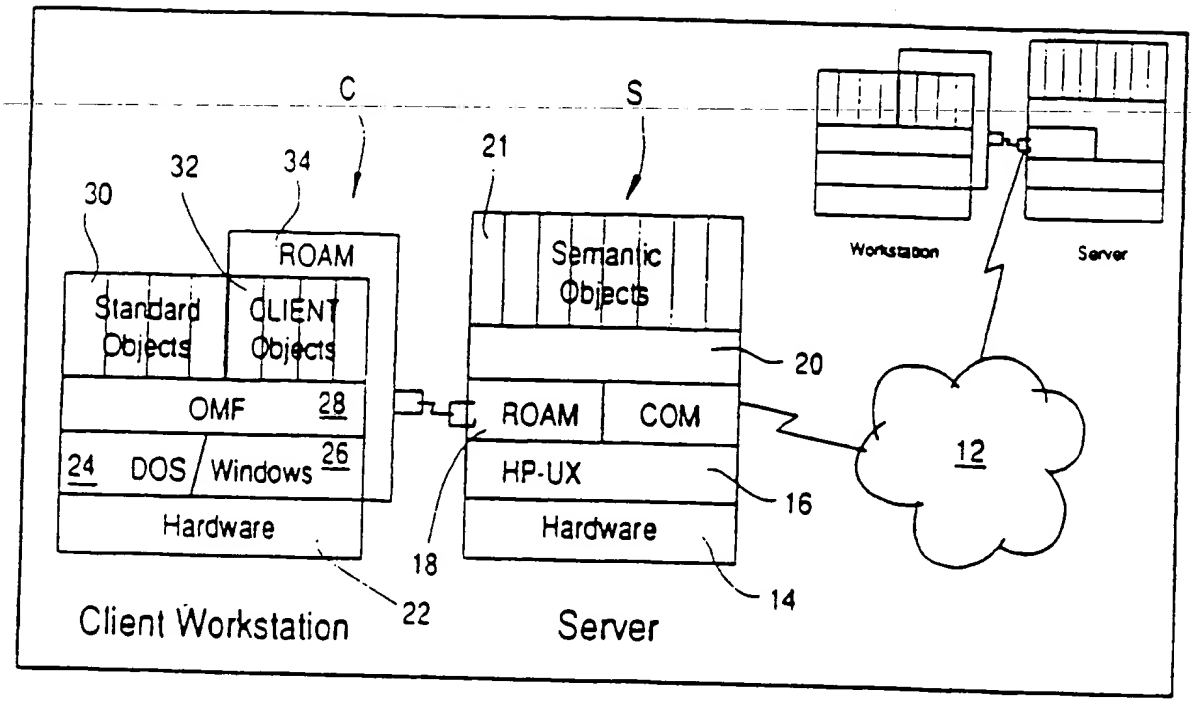

C
S
32
34
21
30
ROAM
Semantic Objects
Standard Objects
CLIENT Objects
20
OMF
28
ROAM
COM
24
DOS
Windows
26
HP-UX
16
12
Hardware
Hardware
18
14
22
Client Workstation
Server
Workstation
Server

FIG 2

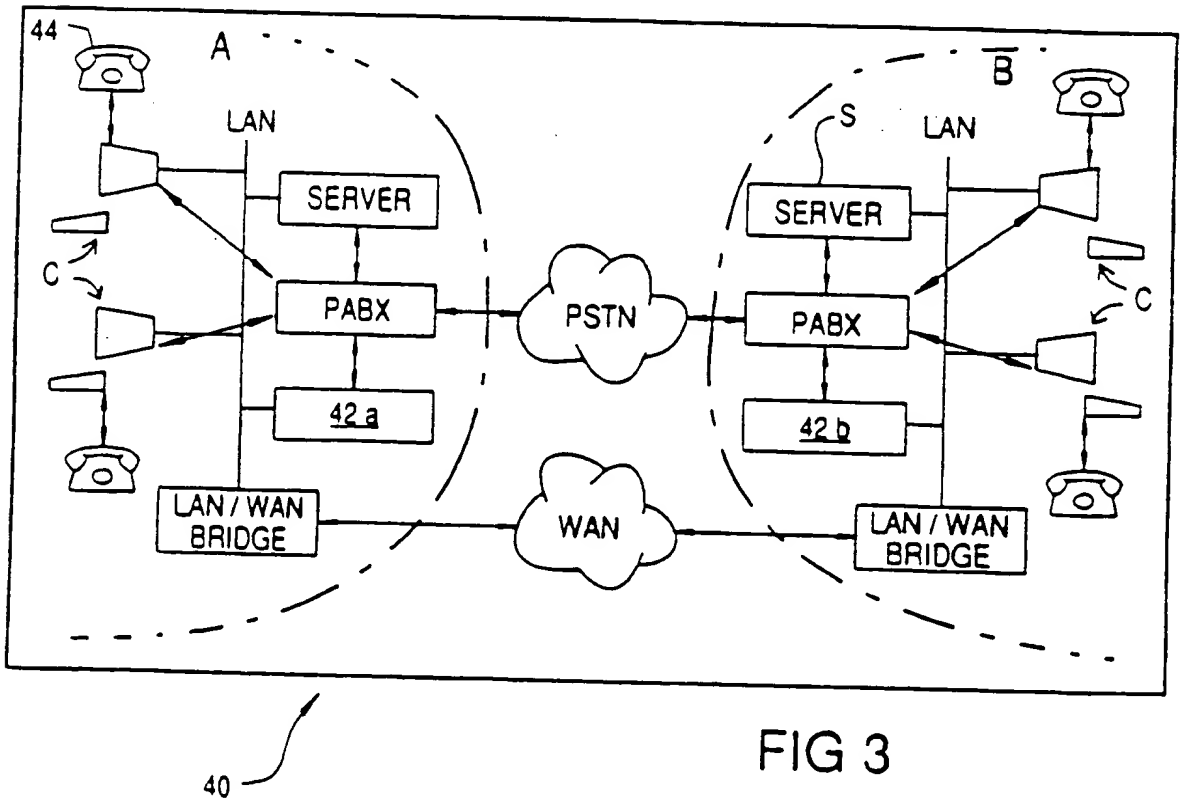

44
A
B
LAN
S
LAN
SERVER
SERVER
C
C
PABX
PSTN
PABX
42 a
42 b
LAN / WAN BRIDGE
WAN
LAN / WAN BRIDGE
40

FIG 3

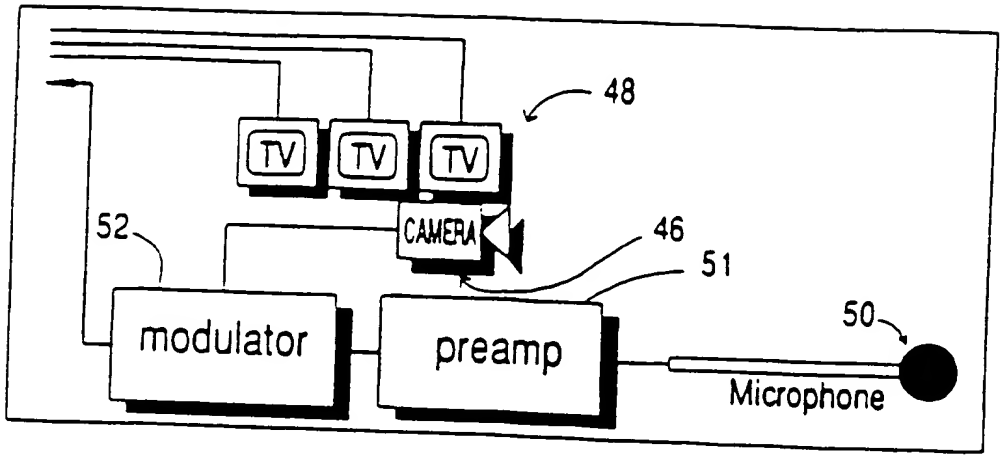

48
TV
TV
TV
CAMERA
52
46
51
modulator
preamp
50
Microphone

FIG 4

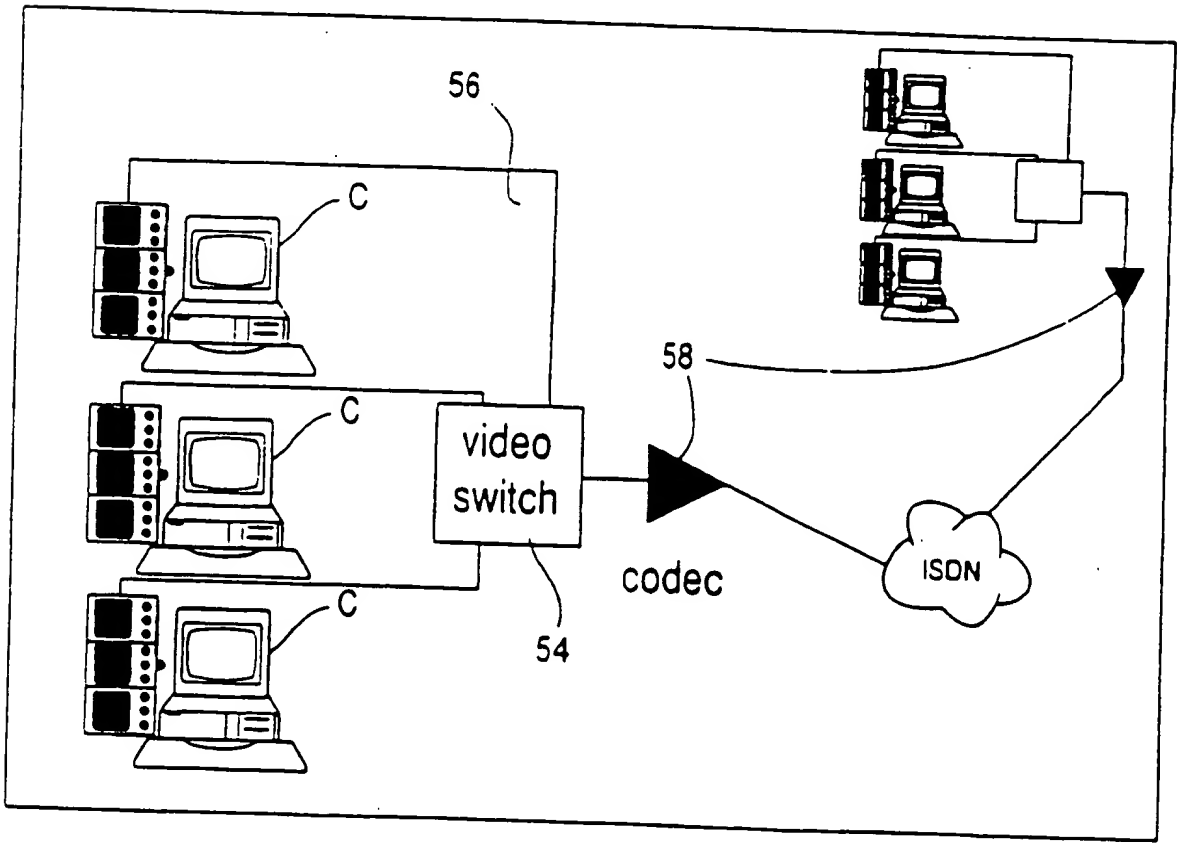

56
C
C
C
video
switch
codec
54
58
ISDN

FIG 5

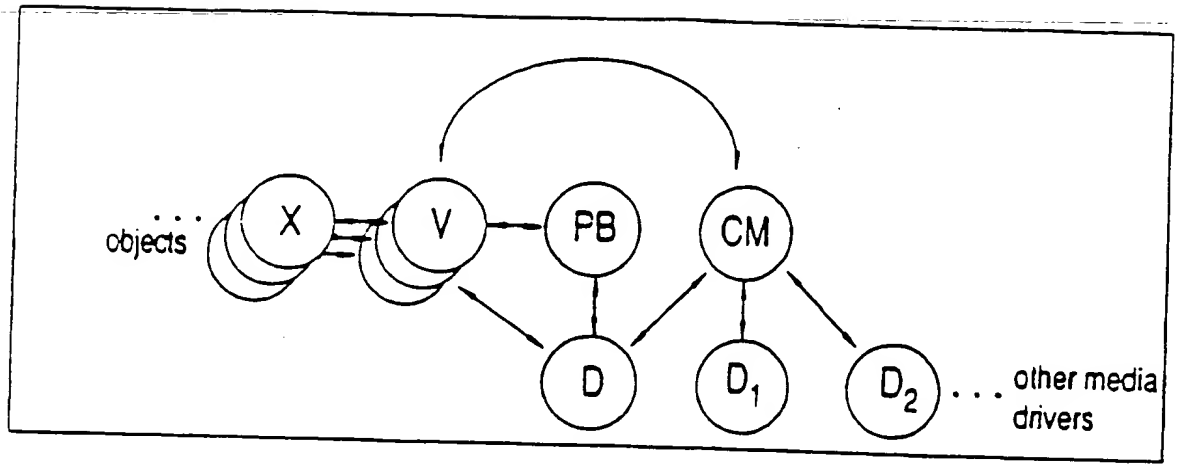

objects
X
V
PB
CM
D
D_1
D_2
other media drivers

FIG 6

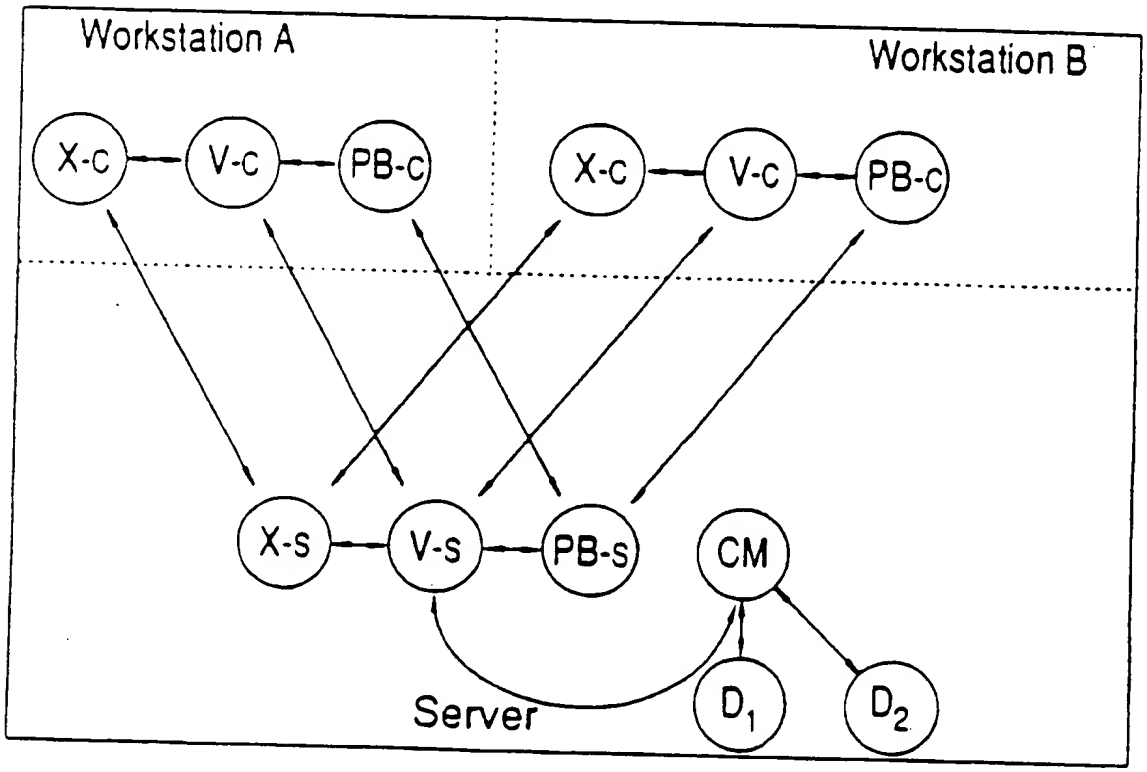

Workstation A
Workstation B
X-C
V-C
PB-C
X-C
V-C
PB-C
X-S
V-S
PB-S
CM
Server
D_1
D_2

FIG 7

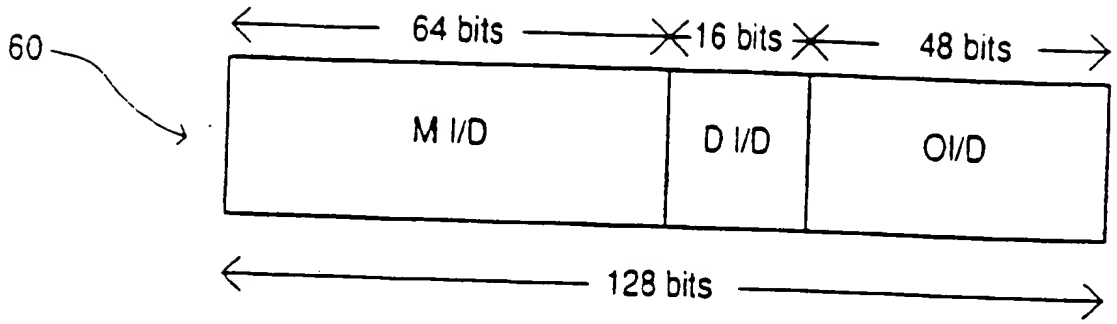


FIG 8

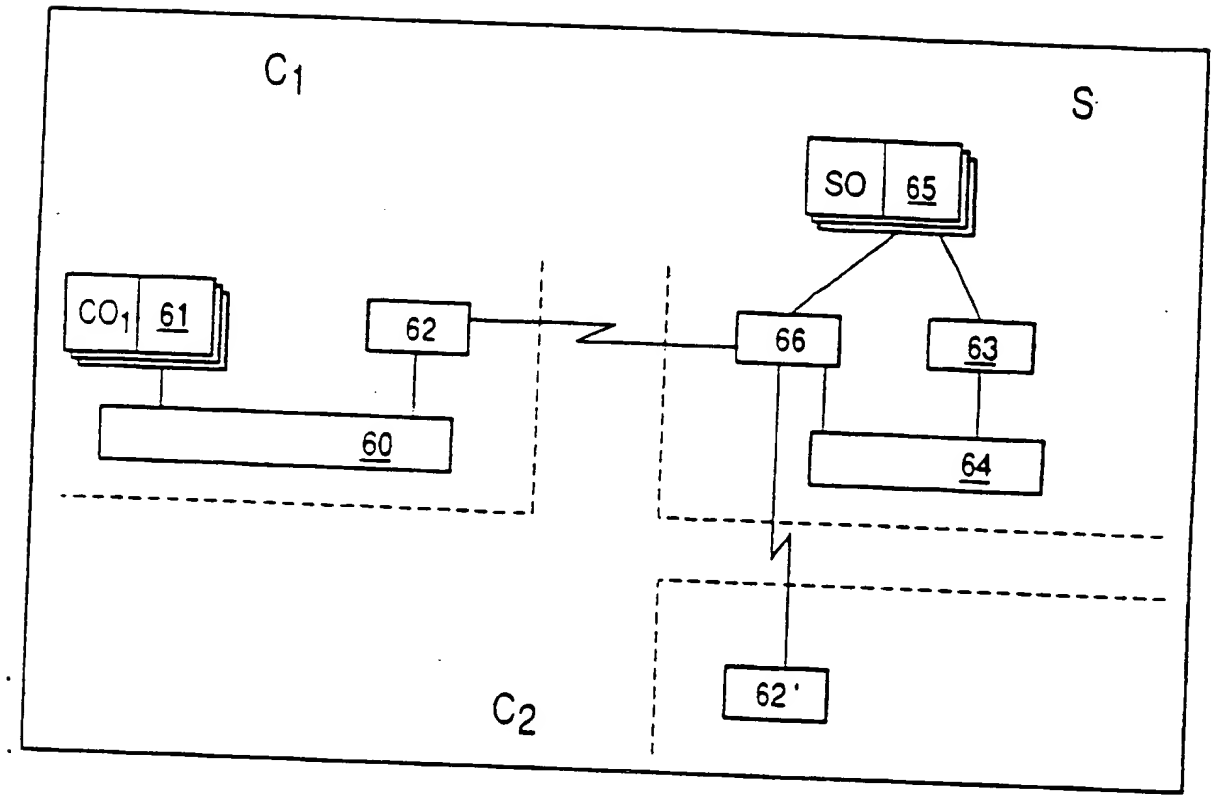


FIG 9

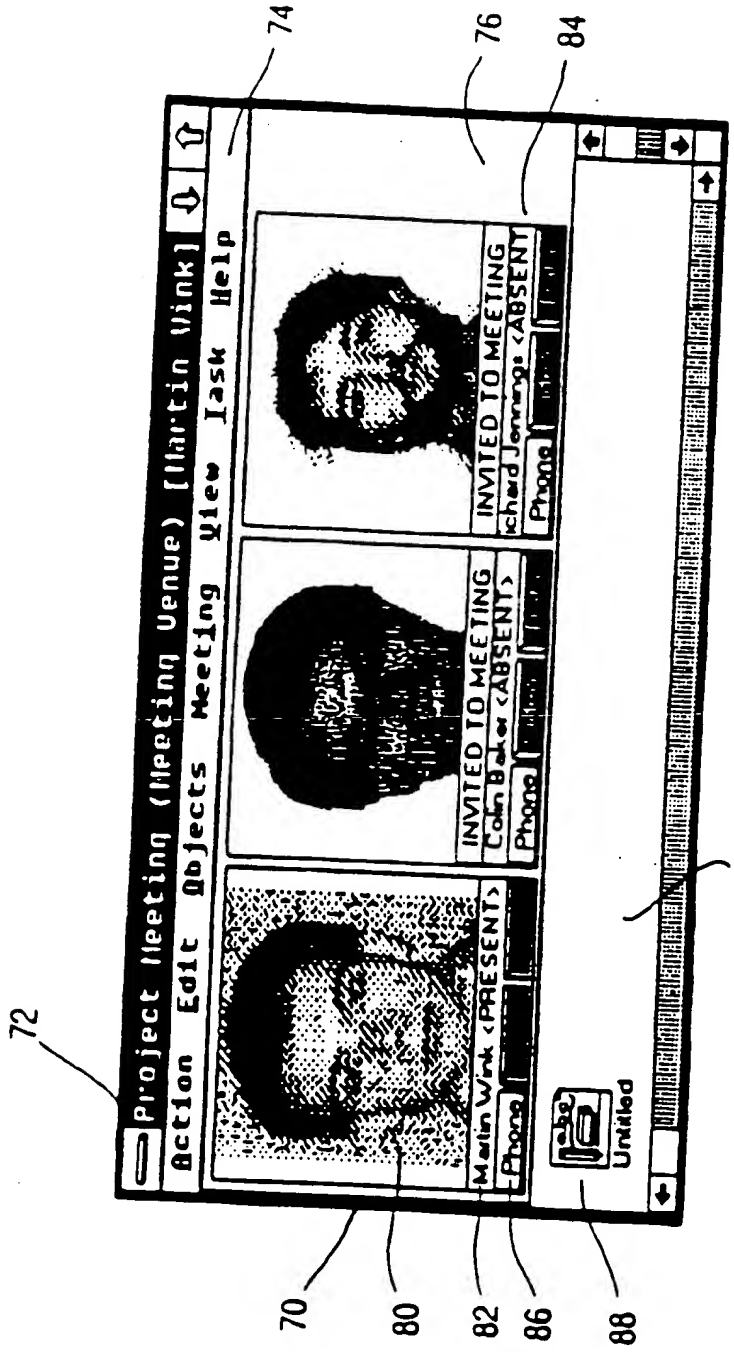

Project Meeting (Meeting Venue) [Martin Wink]
Action Edit Objects Meeting View Task Help
INVITED TO MEETING
Martin Wink <PRESENT>
Phone
INVITED TO MEETING
Colin Baker <ABSENT>
Phone
INVITED TO MEETING
Richard Jennings <ABSENT>
Phone
Untitled
72
74
76
78
70
80
82
84
86
88

FIG 10

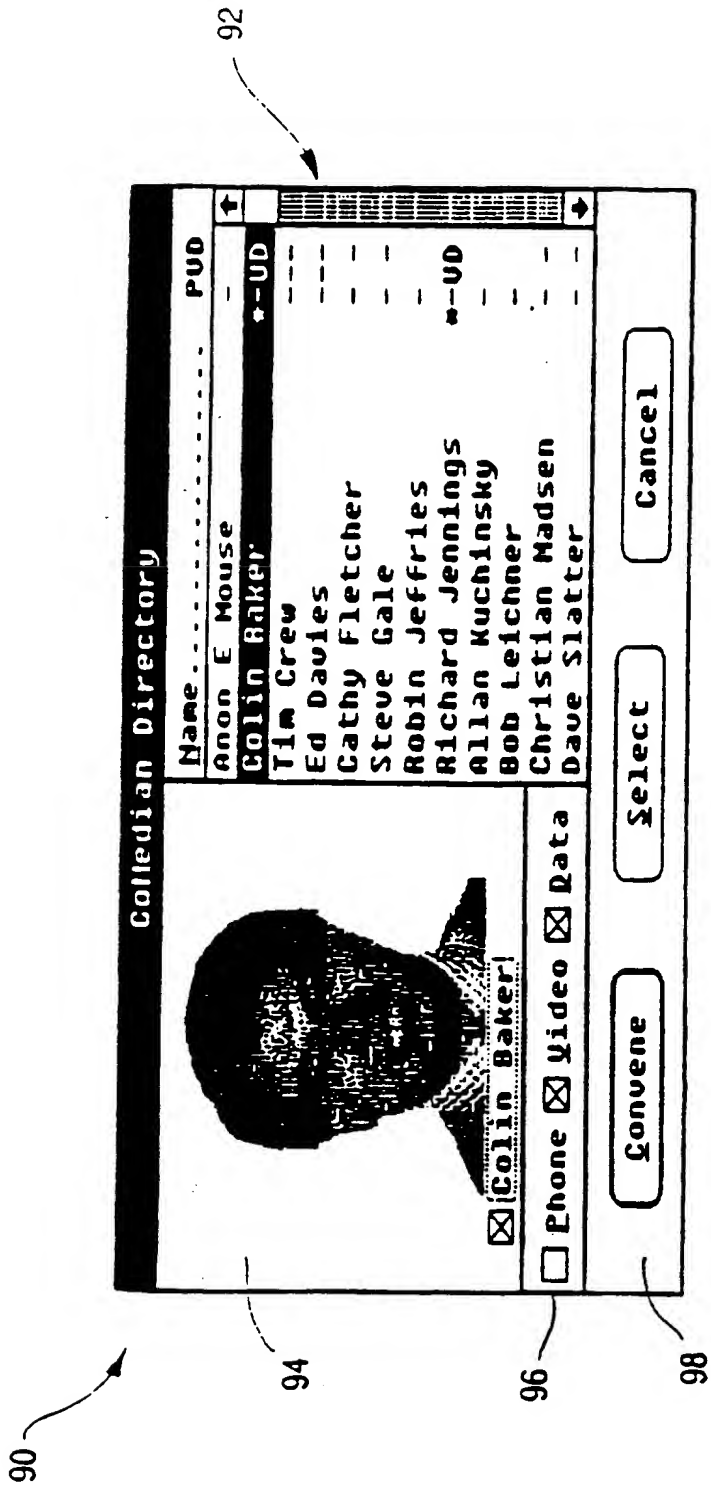

Colledian Directory
Name..... PUD
Anon E Mouse -
Colin Baker *--UD
Tim Crew ---
Ed Davies ---
Cathy Fletcher ---
Steve Gale ---
Robin Jeffries -
Richard Jennings *--UD
Allan Kuchinsky ---
Bob Leichner ---
Christian Madsen ---
Dave Slatter ---
☒Colin Baker!
☐ Phone ☒ Video ☒ Data
Convene
Select
Cancel
90
92
94
96
98

FIG 11

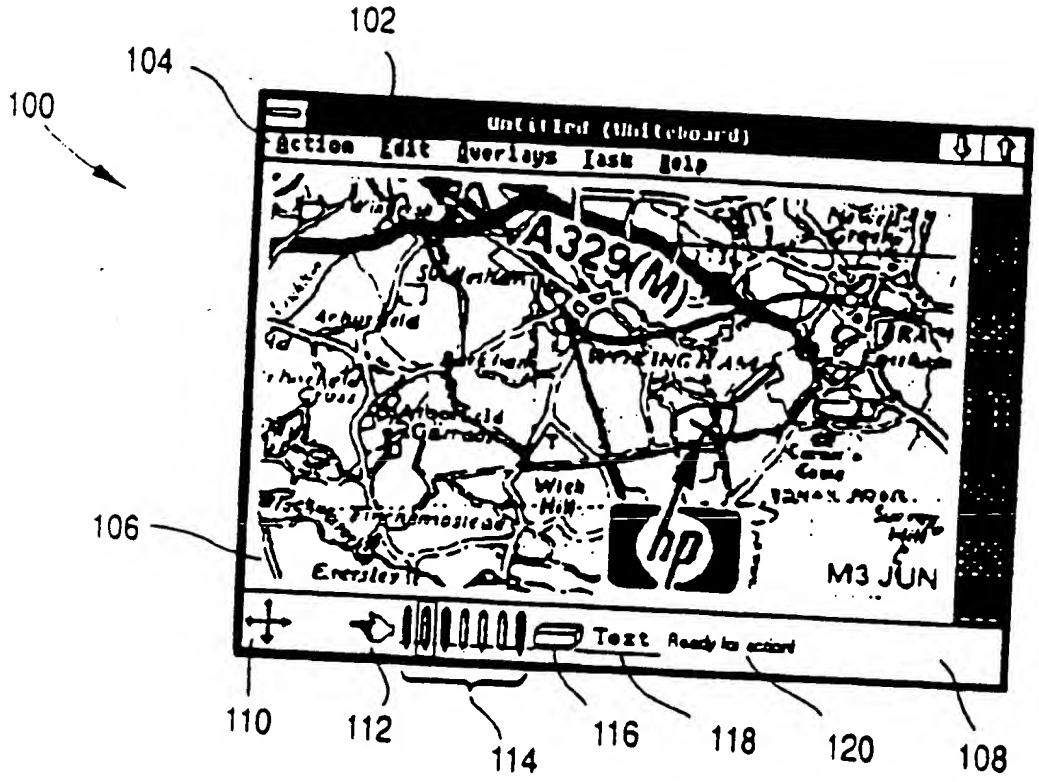

102
104
100
Untitled (Whiteboard)
Action Edit Overlays Task Help
A329(M)
BIRMINGHAM
Wick Hill
Eversley
M3 JUN
hp
Text
Ready to accept
106
110
112
114
116
118
120
108

FIG 12

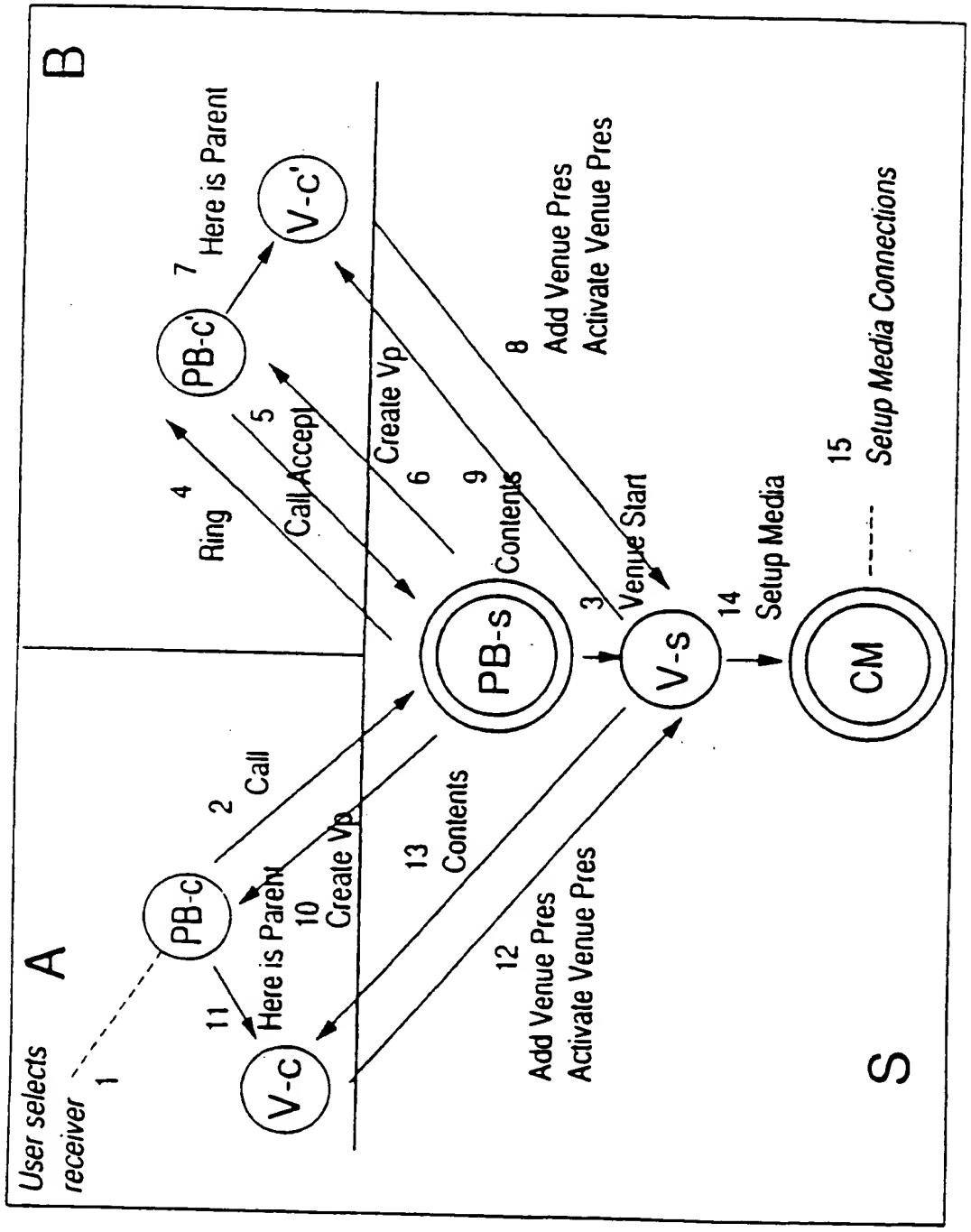

A
B
S
User selects receiver
1
PB-c
2 Call
V-c
11 Here is Parent
10 Create Vp
PB-c'
4 Ring
7 Here is Parent
V-c'
5 Call Accept
6 Create Vp
PB-S
9 Contents
13 Contents
12 Add Venue Pres Activate Venue Pres
8 Add Venue Pres Activate Venue Pres
3 Venue Start
V-S
14 Setup Media
CM
15 Setup Media Connections

FIG 13

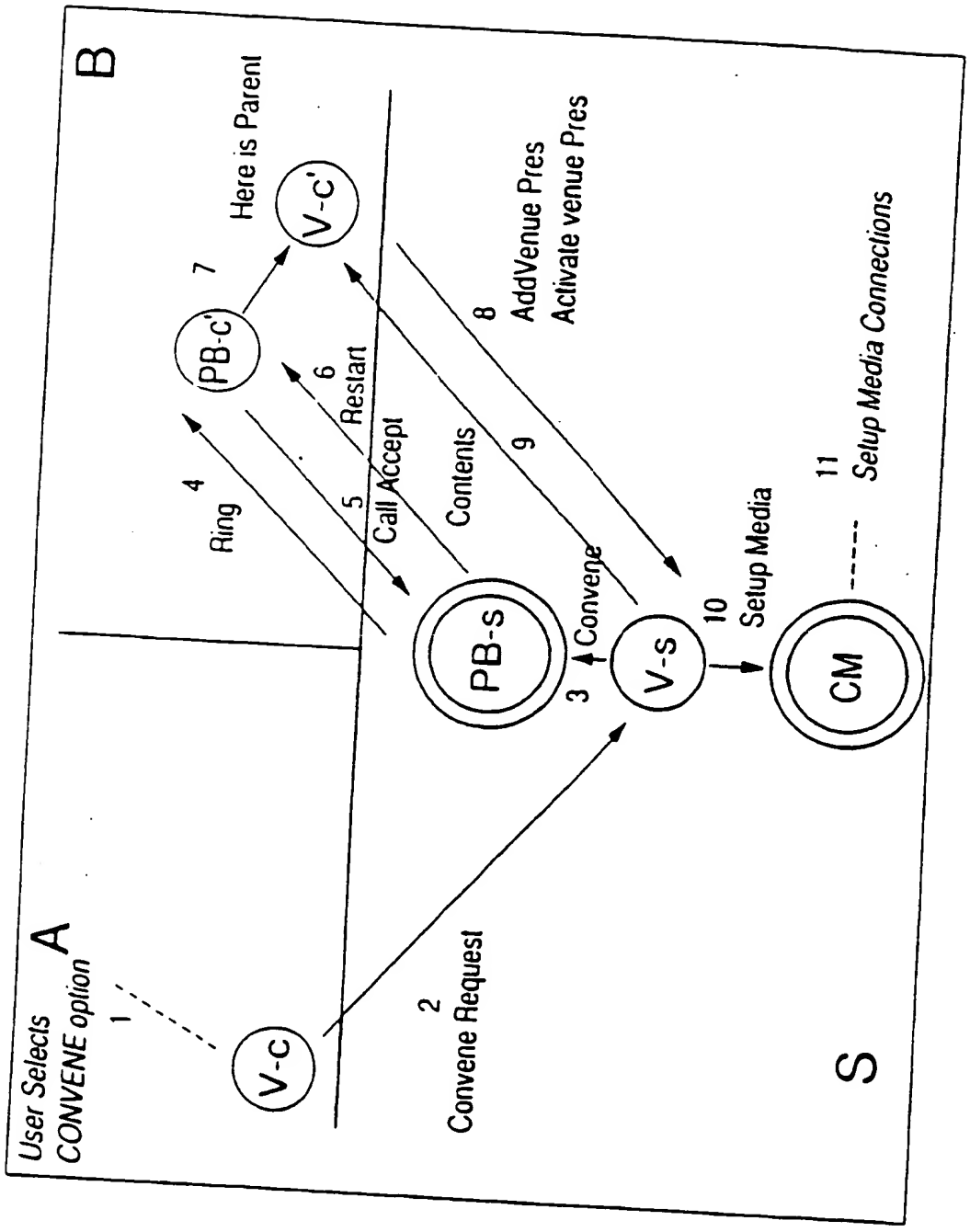

A
B
S
User Selects
CONVENE option
1
V-C
2
Convene Request
PB-S
3
Convene
V-S
4
Ring
PB-C'
5
Call Accept
6
Restart
7
Here is Parent
V-C'
Contents
8
AddVenue Pres
Activate venue Pres
9
10
Setup Media
CM
11
Setup Media Connections

FIG 14

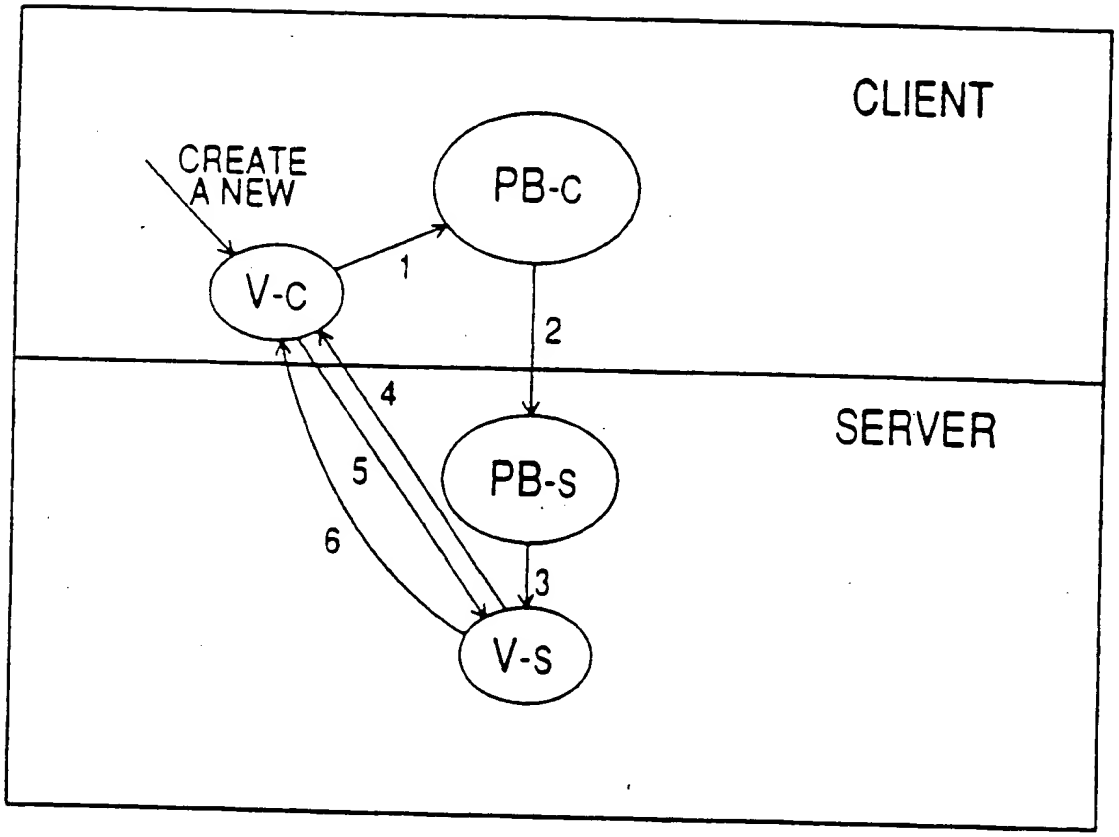


FIG 15

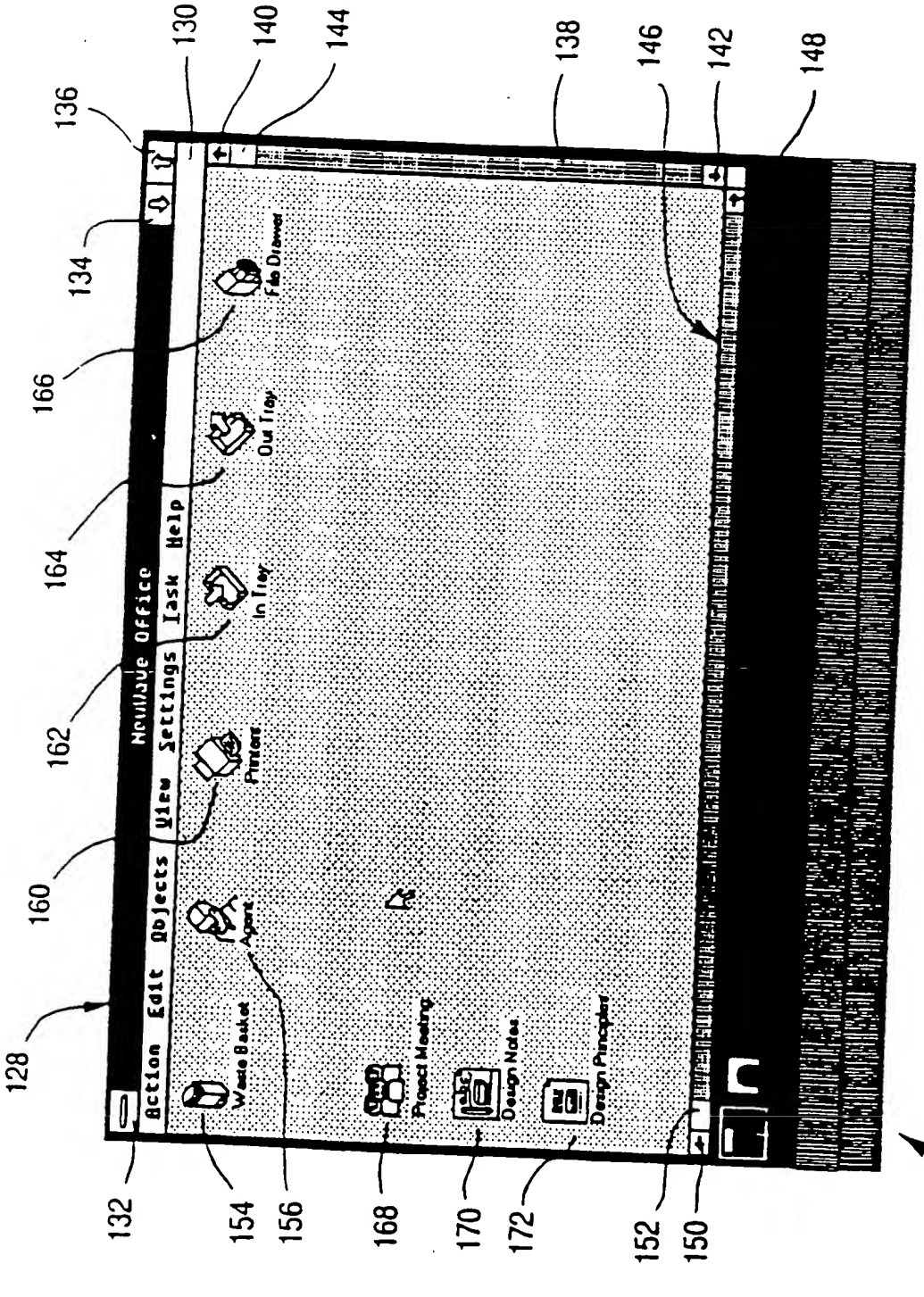

NewWave Office
Action Edit Objects View Settings Task Help
Waste Basket
Agent
Printer
In Tray
Out Tray
File Drawer
Project Meeting
Design Notes
Design Principles
126
128
130
132
134
136
138
140
142
144
146
148
150
152
154
156
160
162
164
166
168
170
172

FIG 16

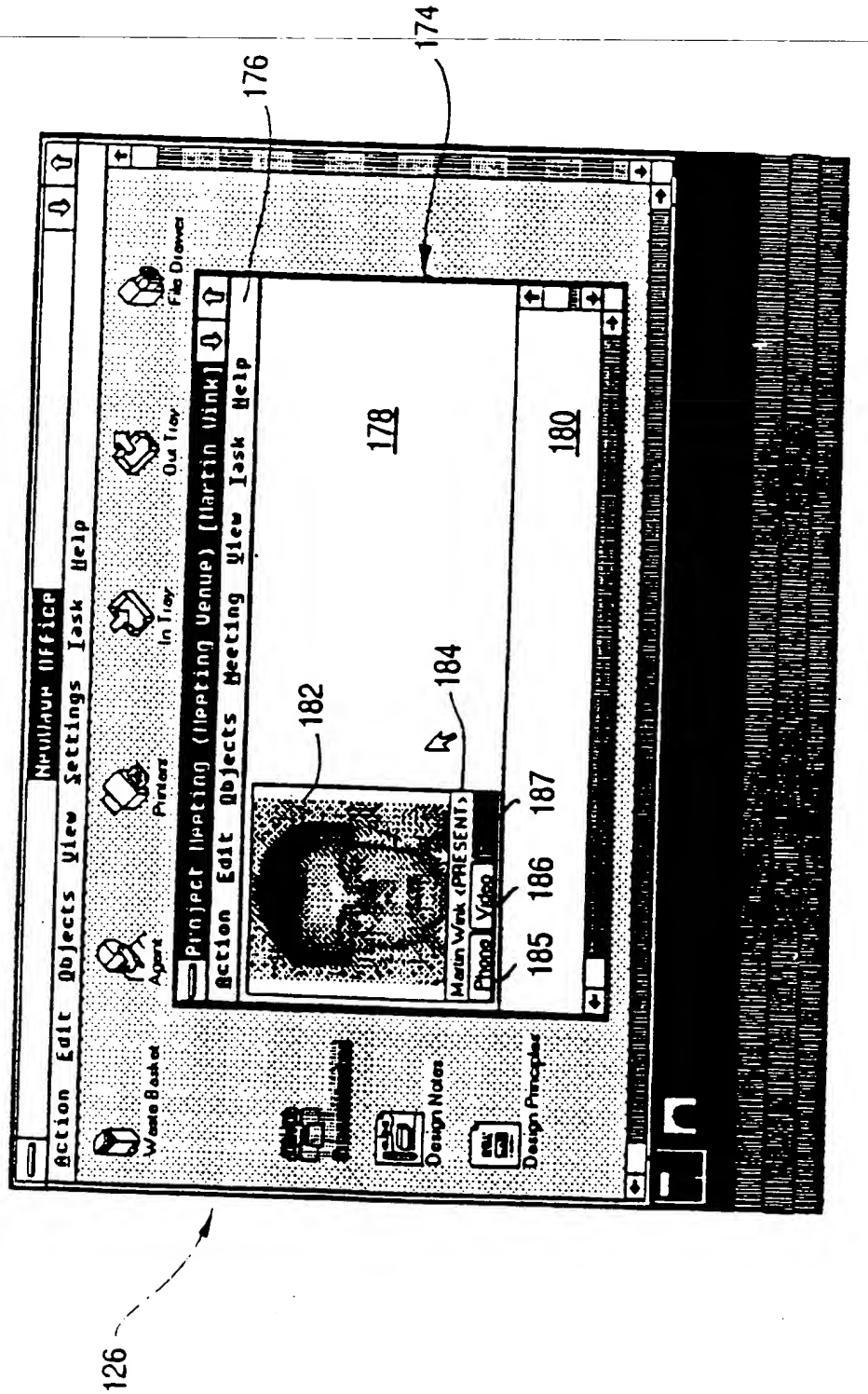

174
176
178
180
182
184
185
186
187
126
Action Edit Objects View Settings Task Help
Project Meeting (Meeting Venue) [Martin Wink]
Action Edit Objects Meeting View Task Help
Martin Wink (PRESENT)
Phone
Video
Waste Basket
Agent
Printer
In Tray
Out Tray
File Drawer
Design Notes
Design Principles

FIG 17

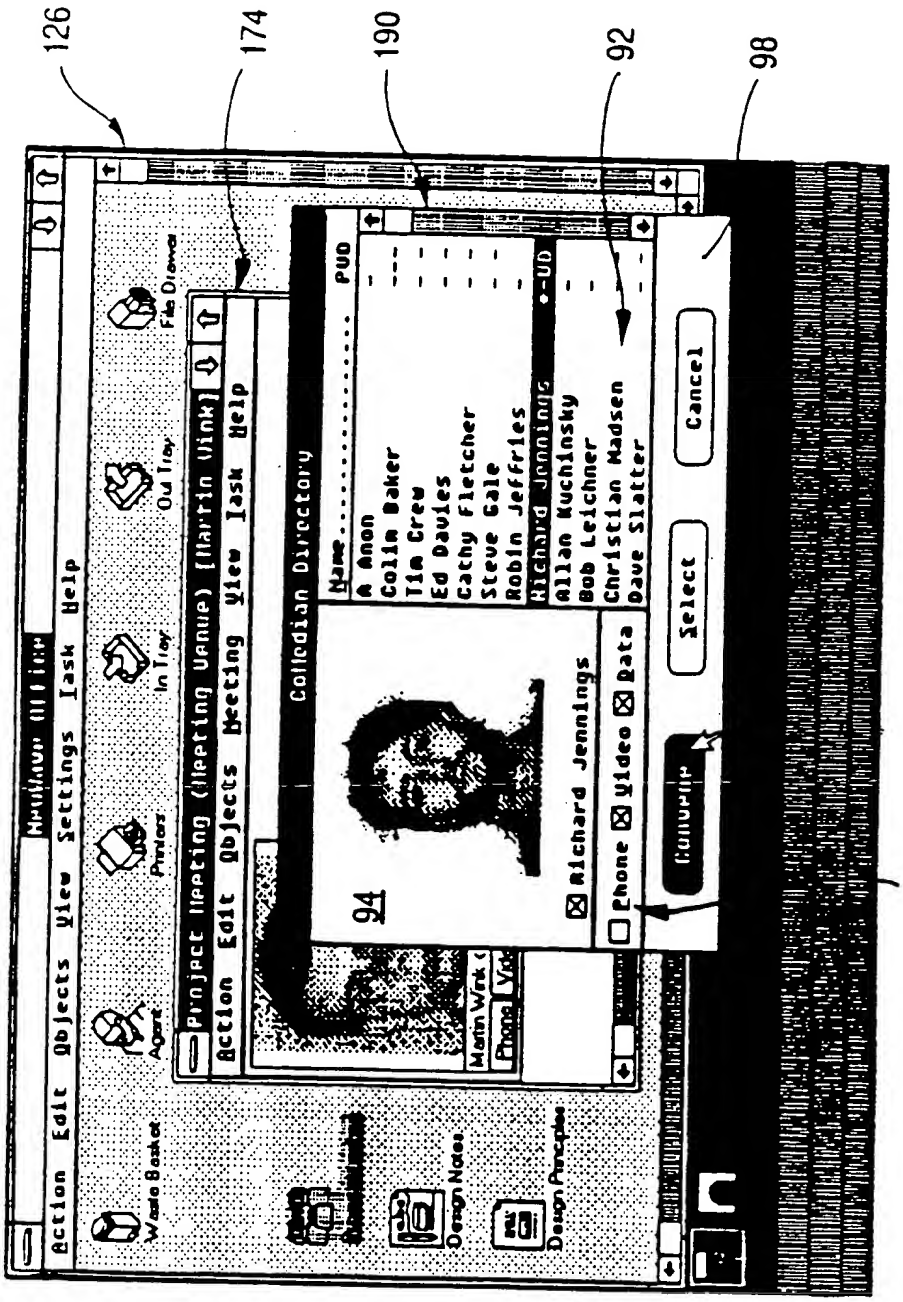

126
174
190
92
98
96
94
Action Edit Objects View Settings Task Help
Project Meeting (Meeting Venue) [Martin Wink]
Action Edit Objects Meeting View Task Help
Collegian Directory
Name
A Anon
Colin Baker
Tim Crew
Ed Davies
Cathy Fletcher
Steve Gale
Robin Jeffries
Richard Jennings
Allan Kuchinsky
Bob Leichner
Christian Madsen
Dave Slatter
Richard Jennings
Phone Video Data
Select
Cancel
Waste Basket
Agent
Printers
Out Tray
In Tray
File Drawer
Design Notes
Design Principles

FIG 18

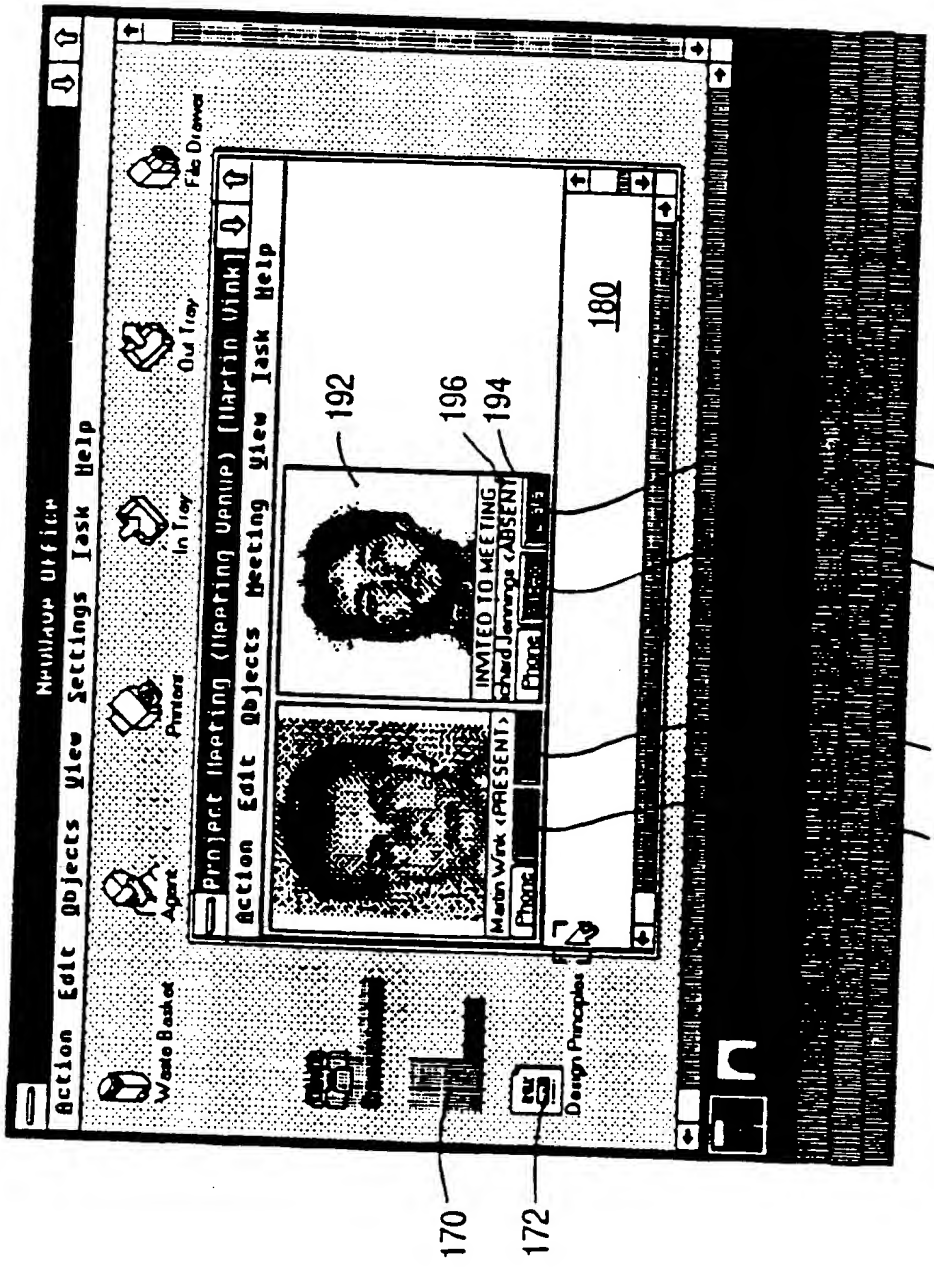

Action Edit Objects View Settings Task Help
Waste Basket
Agent
Printer
In Tray
Out Tray
File Drawer
Project Meeting (Meeting Online) [Martin Vink]
Action Edit Objects Meeting View Task Help
192
196
194
180
INVITED TO MEETING
Chad Jennings (ABSENT)
Phone
Main Work (PRESENT)
Phone
Design Principles
170
172
186
187
186a
187a

FIG19

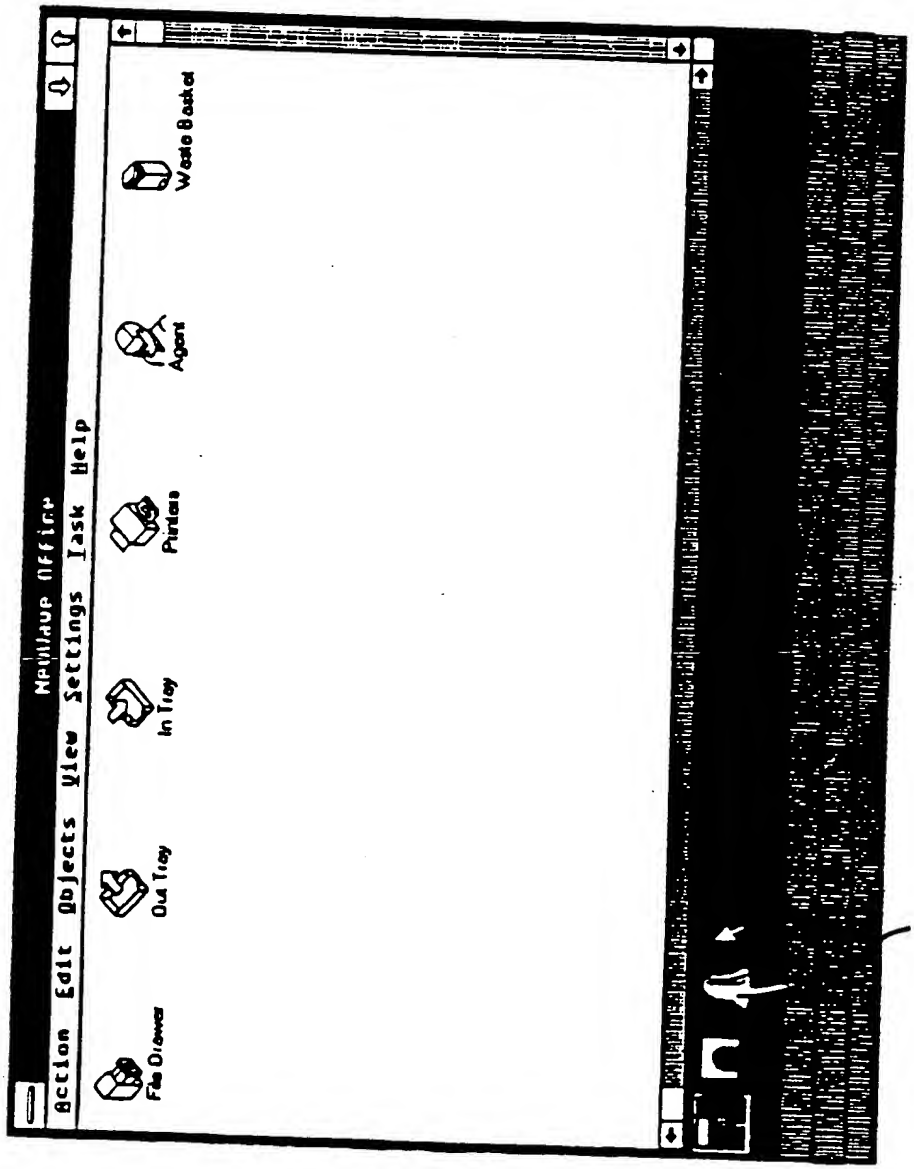

Action Edit Objects View Settings Task Help
File Drawer
Out Tray
In Tray
Printers
Agent
Waste Basket
200

FIG 20

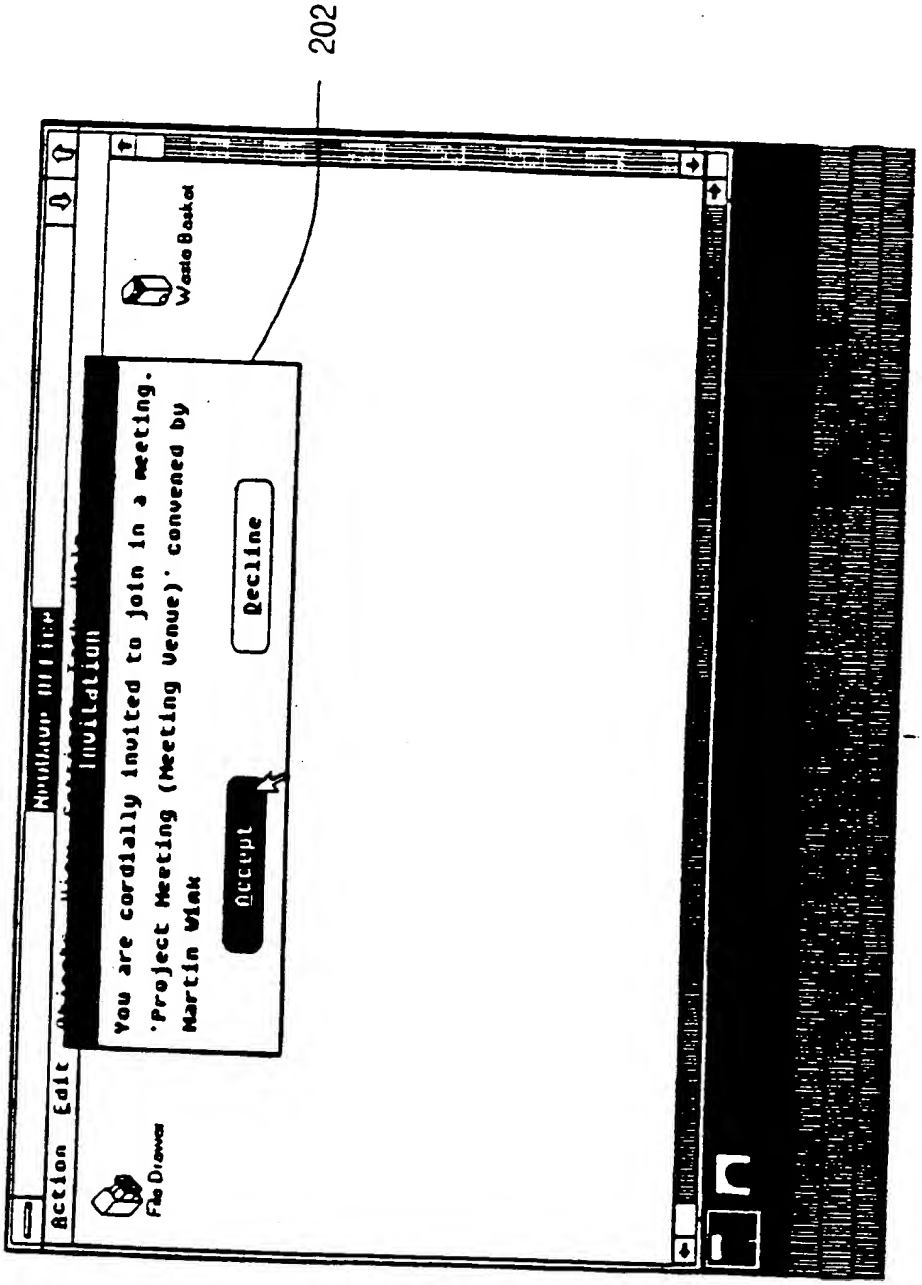

202
Action Edit
File Drawer
Waste Basket
Invitation
You are cordially invited to join in a meeting.
'Project Meeting (Meeting Venue)' convened by
Martin Wiek
Accept
Decline

FIG 21

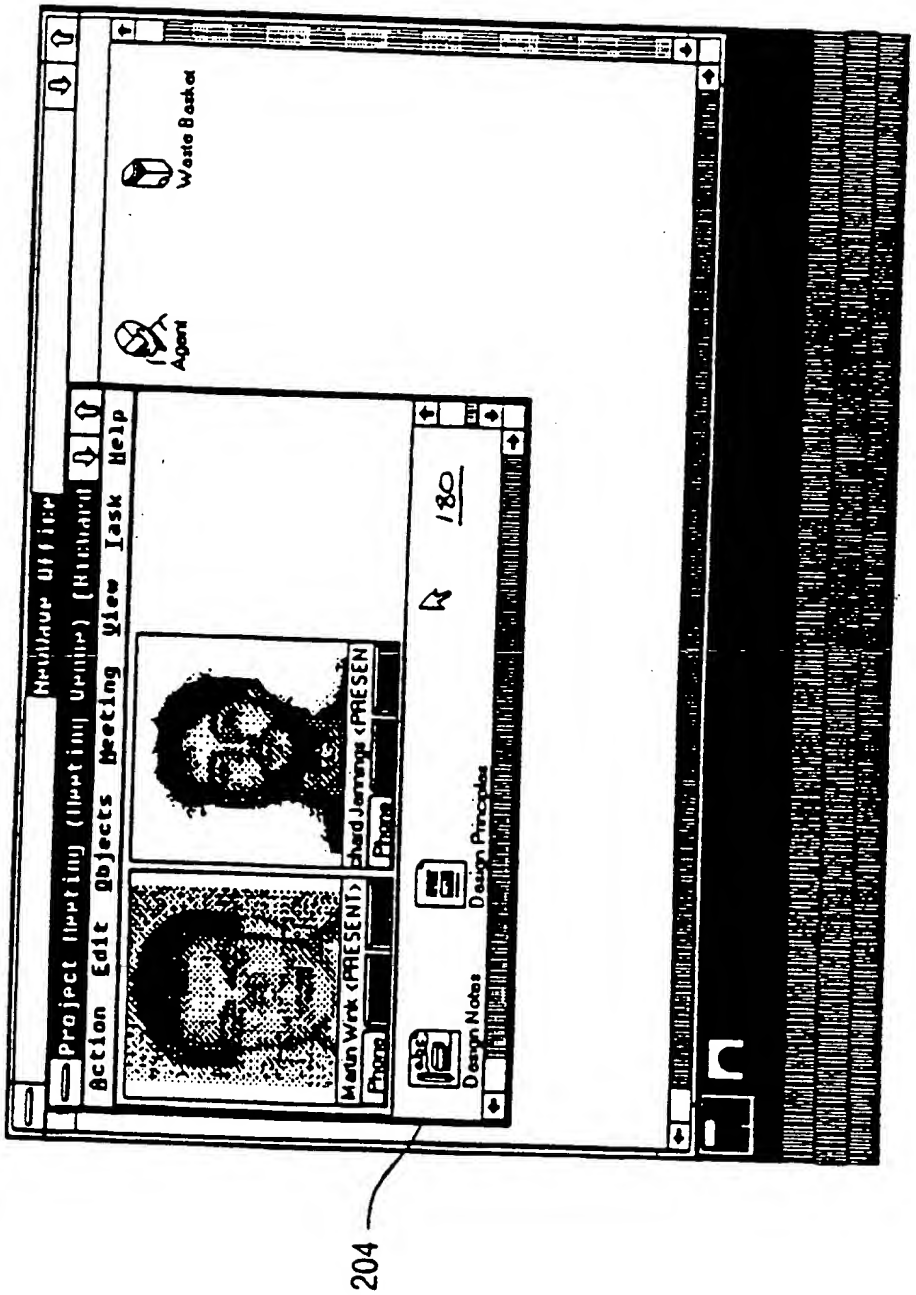

Waste Basket
Agent
Action Edit Objects Meeting View Task Help
Martin Wrtk (PRESENT)
Phone
Design Notes
Design Principles
180
204

FIG 22

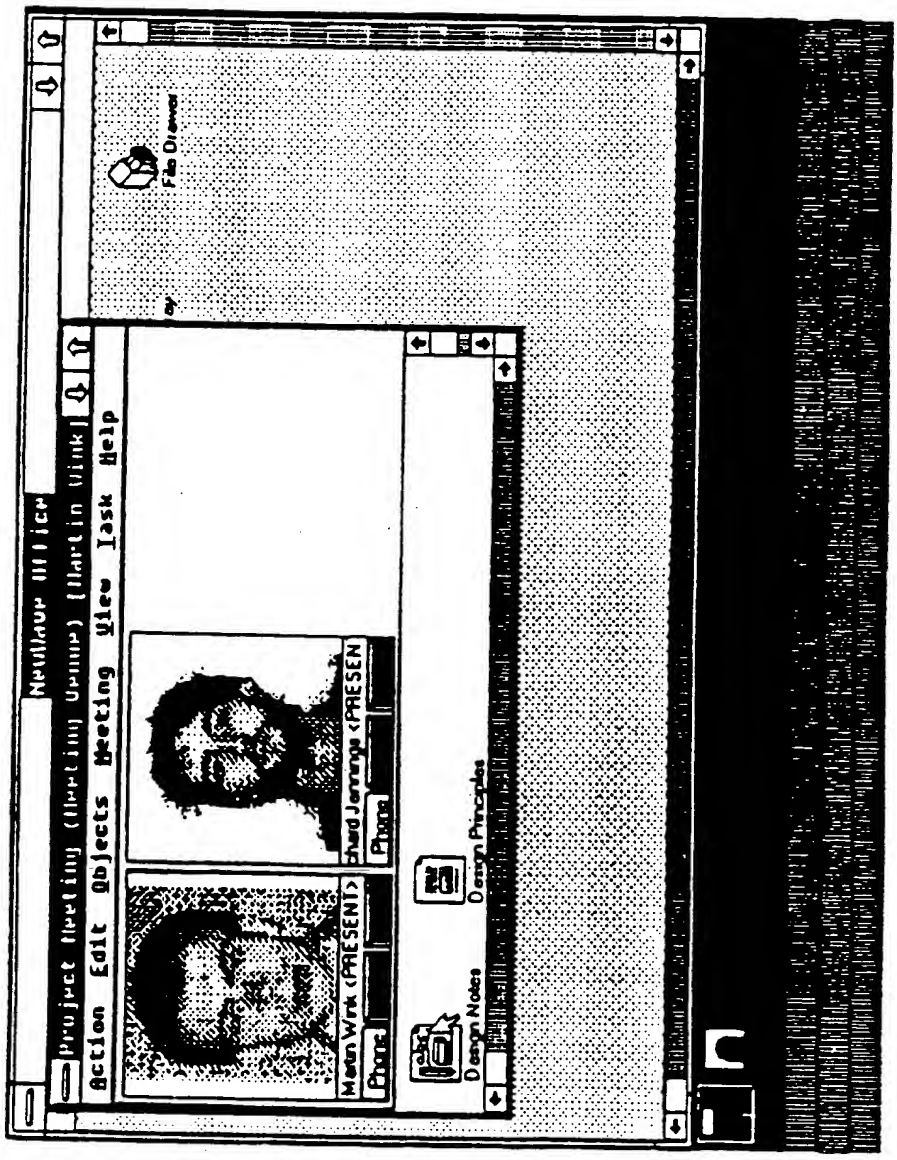

Action Edit Objects Meeting View Task Help
Design Notes
Design Principles
File Drawer

FIG 23

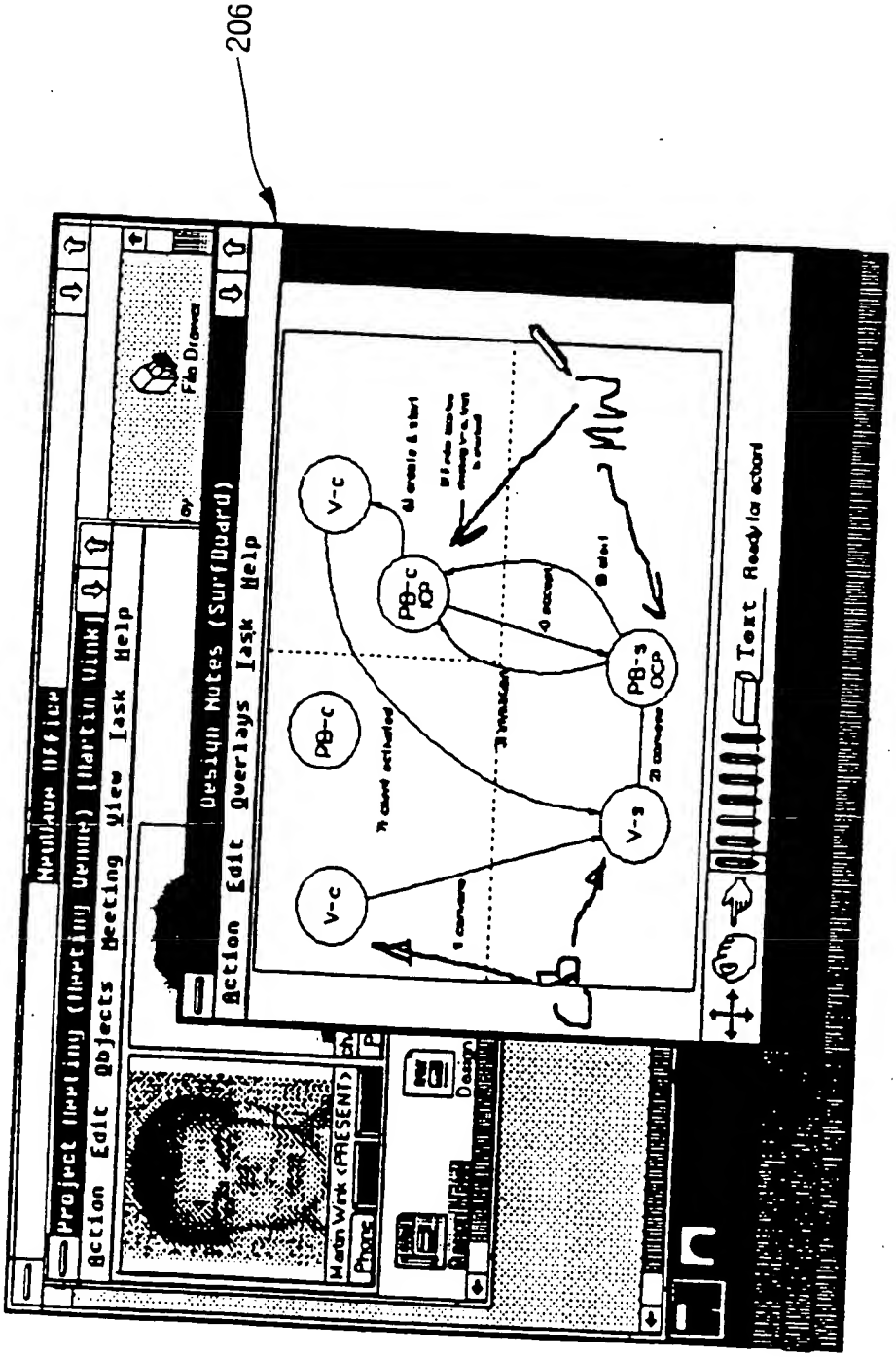

206
Project Meeting (Meeting Venue) (Martin Wink)
Action Edit Objects Meeting View Task Help
Design Notes (SurfBoard)
Action Edit Overlays Task Help
V-C
PB-C
V-S
PB-C OCP
PB-S OCP
Text Ready for action

FIG 24

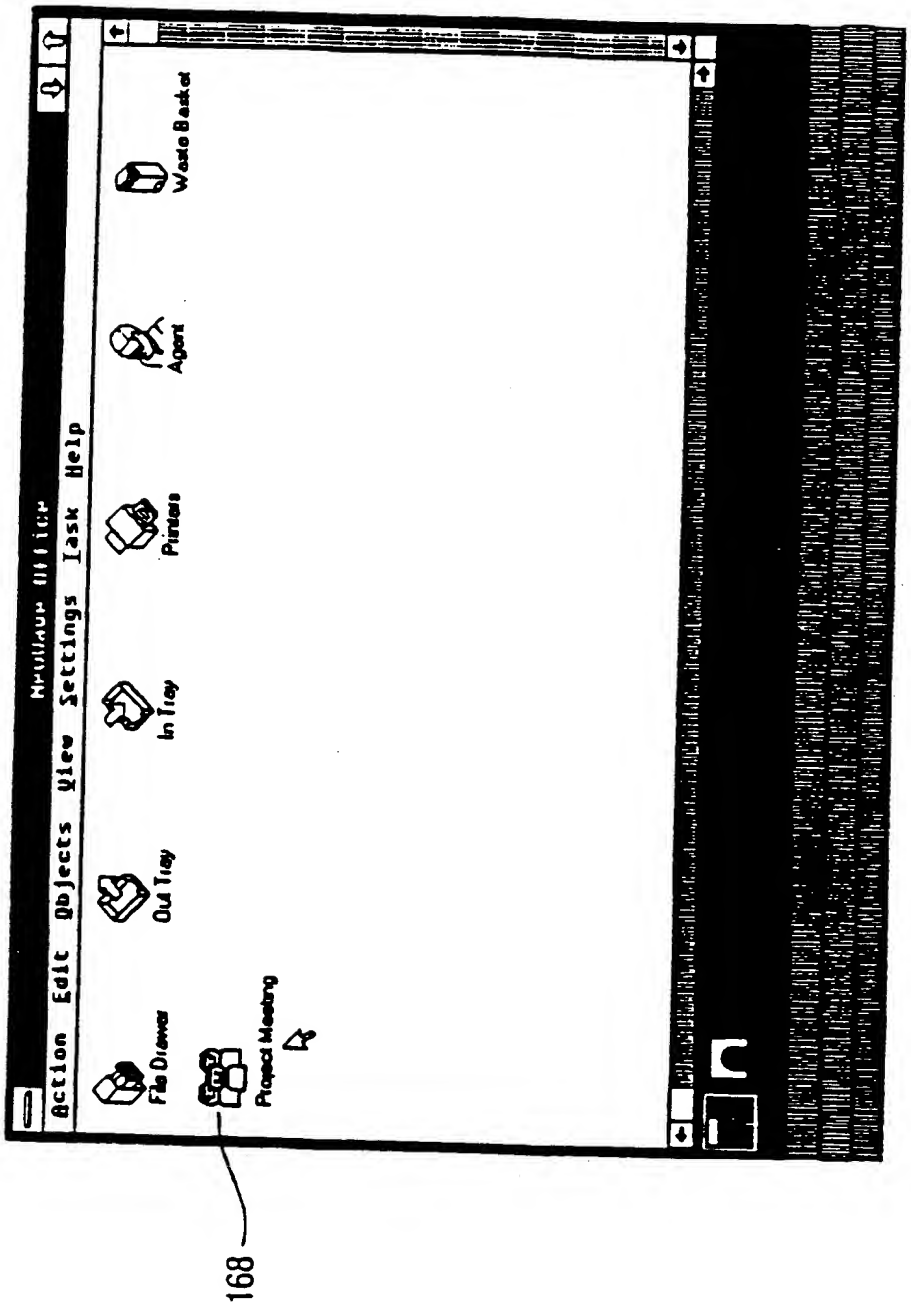

Action Edit Objects View Settings Task Help
File Drawer
Out Tray
In Tray
Printers
Agent
Waste Basket
Project Meeting
168

FIG 25

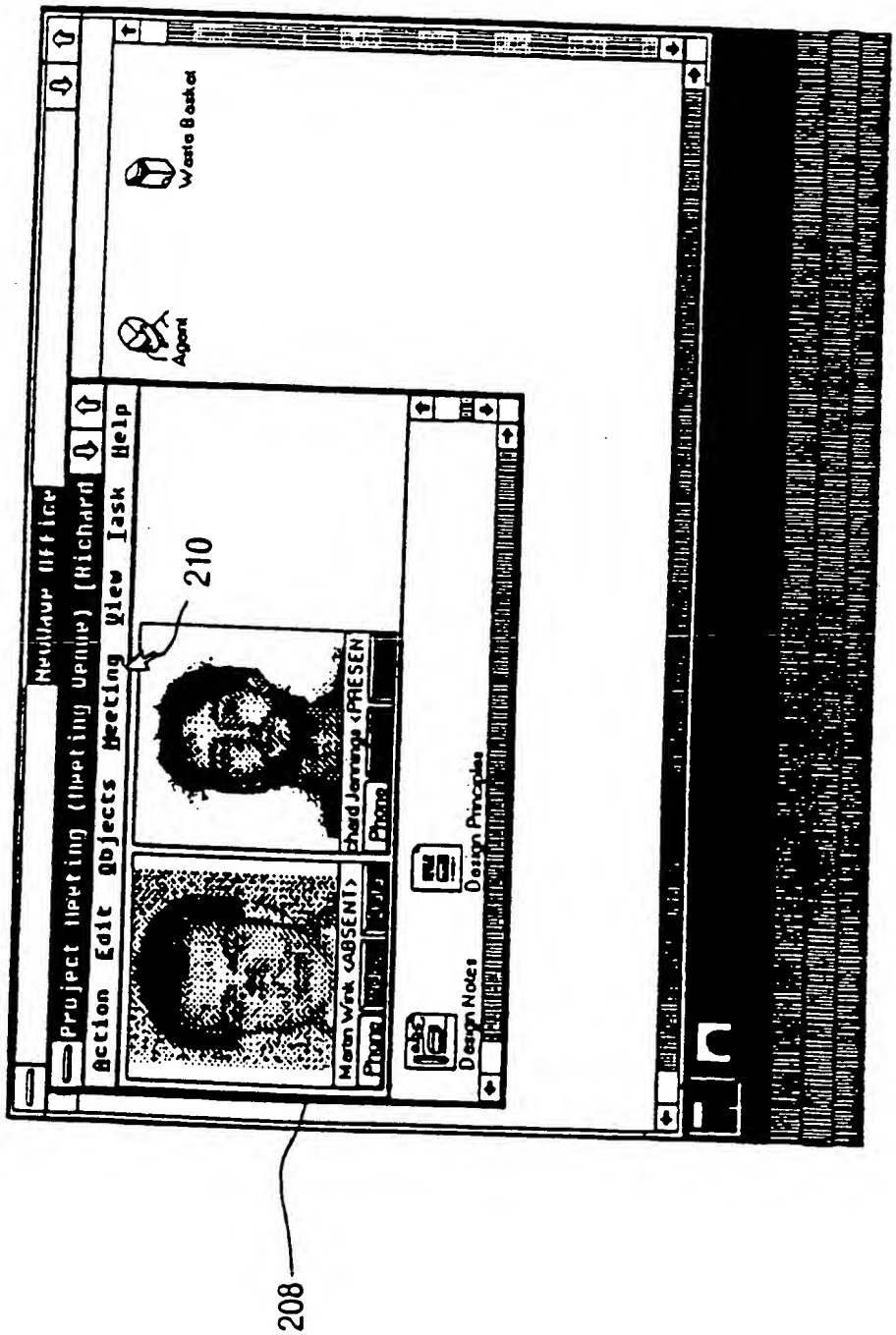

Project Meeting (Meeting Online) (Richard)
Action Edit Objects Meeting View Task Help
Agent
Waste Basket
Design Notes
Design Principles
Phone
210
208

FIG 26

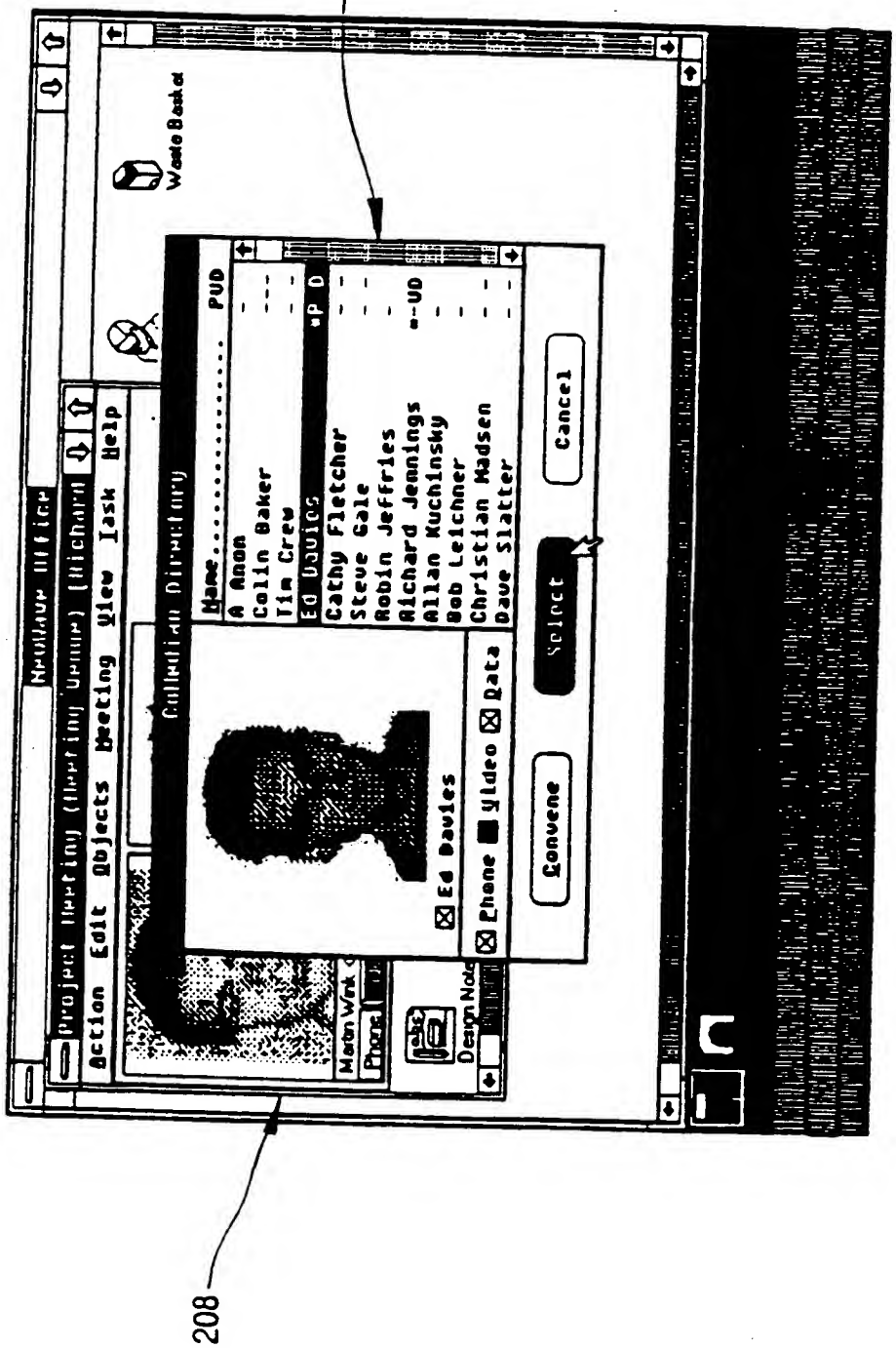

212
208
Waste Basket
Action Edit Objects Meeting View Task Help
Confidential Directory
Name ..... PUB
A Anon
Colin Baker
Tim Crew
Ed Davies
Cathy Fletcher
Steve Gale
Robin Jeffries
Richard Jennings
Allan Kuchinsky
Bob Leichner
Christian Madsen
Dave Slatter
Ed Davies
Phone Video Data
Convene
Select
Cancel

FIG 27

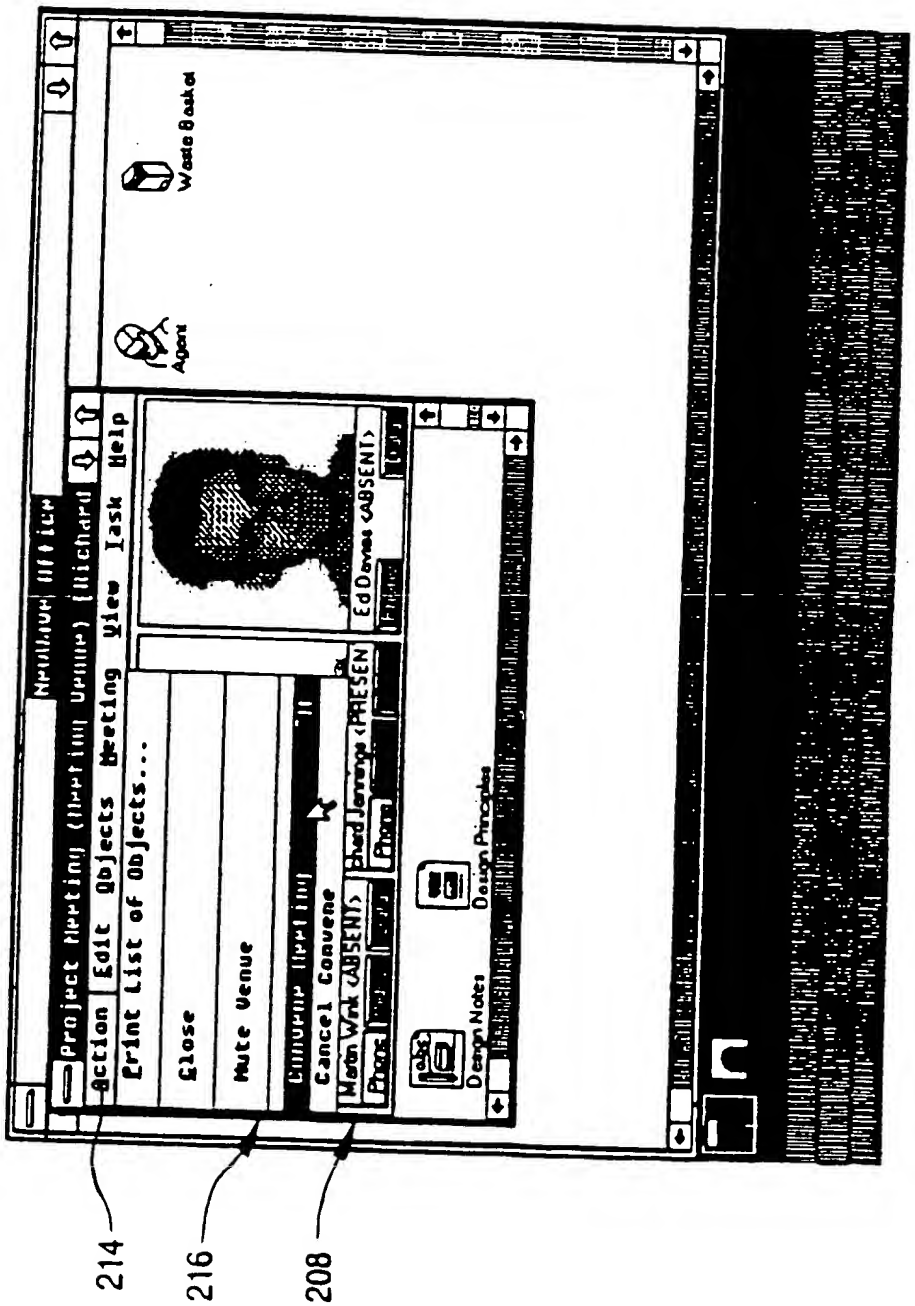

Project Meeting
Action
Edit
Objects
Meeting
View
Task
Help
Print List of Objects...
Close
Mute Venue
Cancel Convene
Main Wnd
Waste Basket
Agent
Design Notes
Design Principles
214
216
208

FIG 28

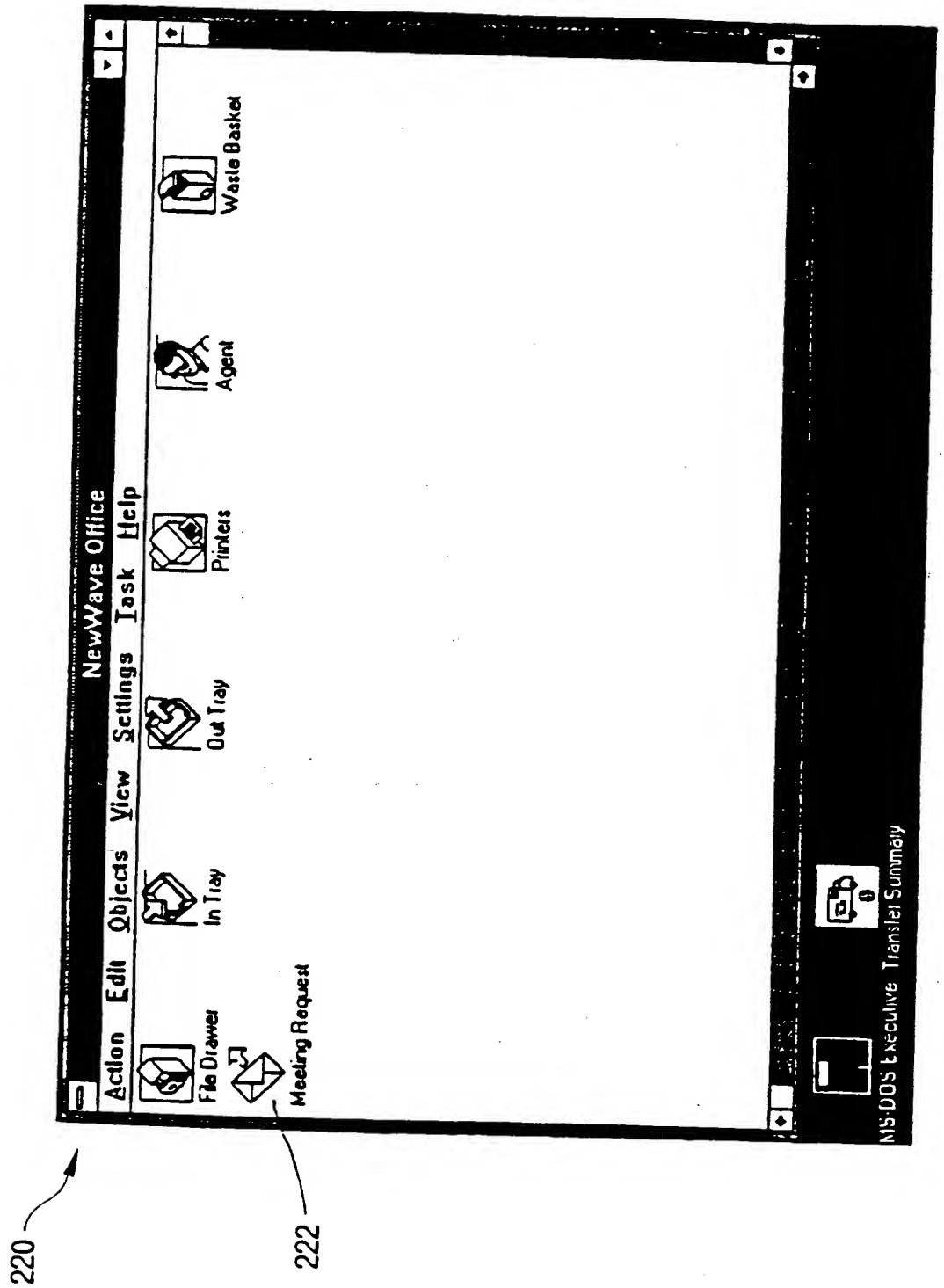

NewWave Office
Action Edit Objects View Settings Task Help
File Drawer
In Tray
Out Tray
Printers
Agent
Waste Basket
Meeting Request
MS-DOS Executive
Transfer Summary
220
222

FIG 29

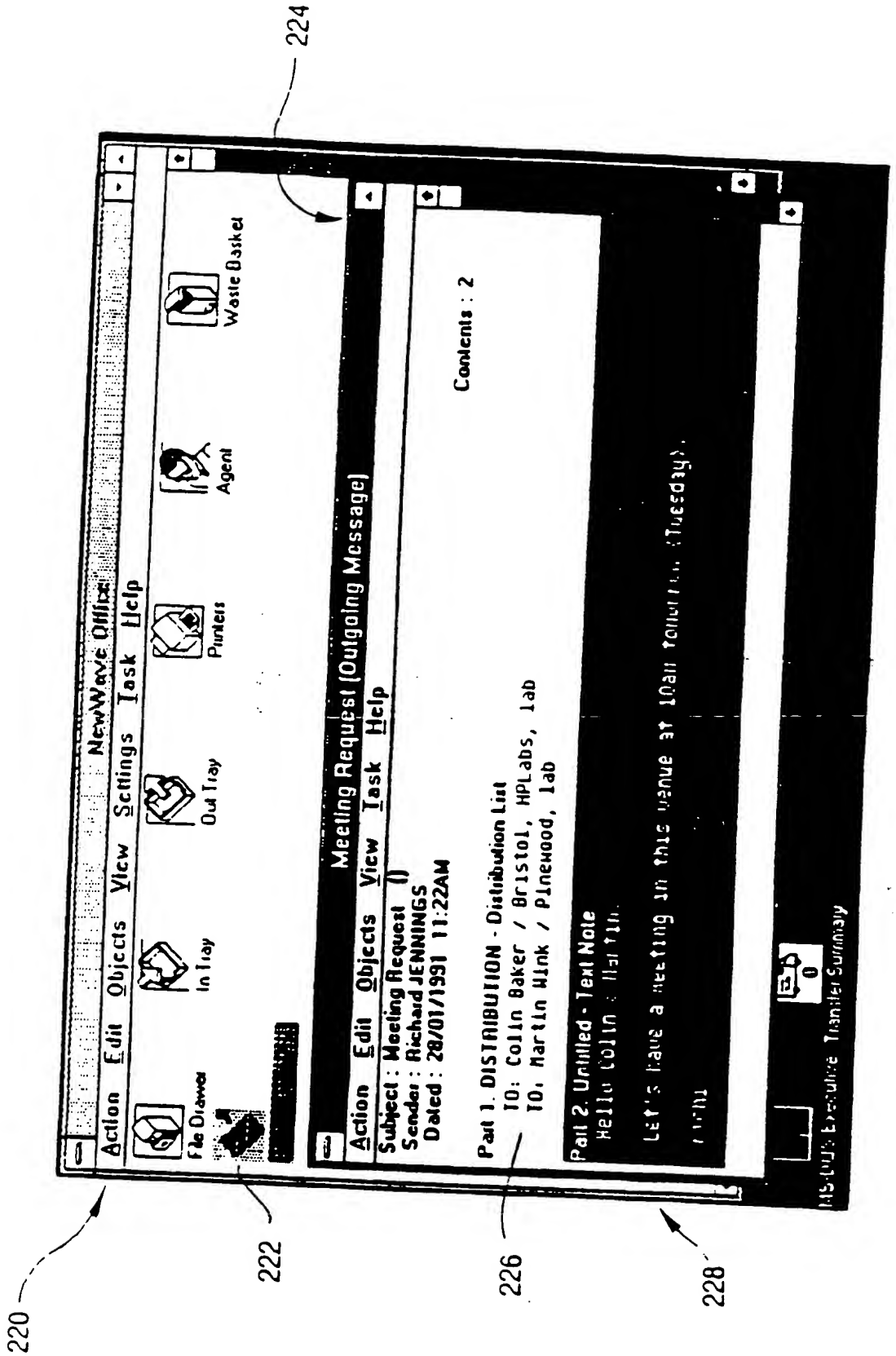

220
222
224
226
228
Action Edit Objects View Settings Task Help
File Drawer
In Tray
Out Tray
Printers
Agent
Waste Basket
Meeting Request (Outgoing Message)
Action Edit Objects View Task Help
Subject : Meeting Request ()
Sender : Richard JENNINGS
Dated : 28/01/1991 11:22AM
Contents : 2
Part 1. DISTRIBUTION - Distribution List
TO: Colin Baker / Bristol, HPLabs, Lab
TO: Martin Wink / Pinewood, Lab
Part 2. Untitled - Text Note
Hello Colin ,
Let's have a meeting in this venue at 10am tomorrow (Tuesday).
Colin
0

FIG 30

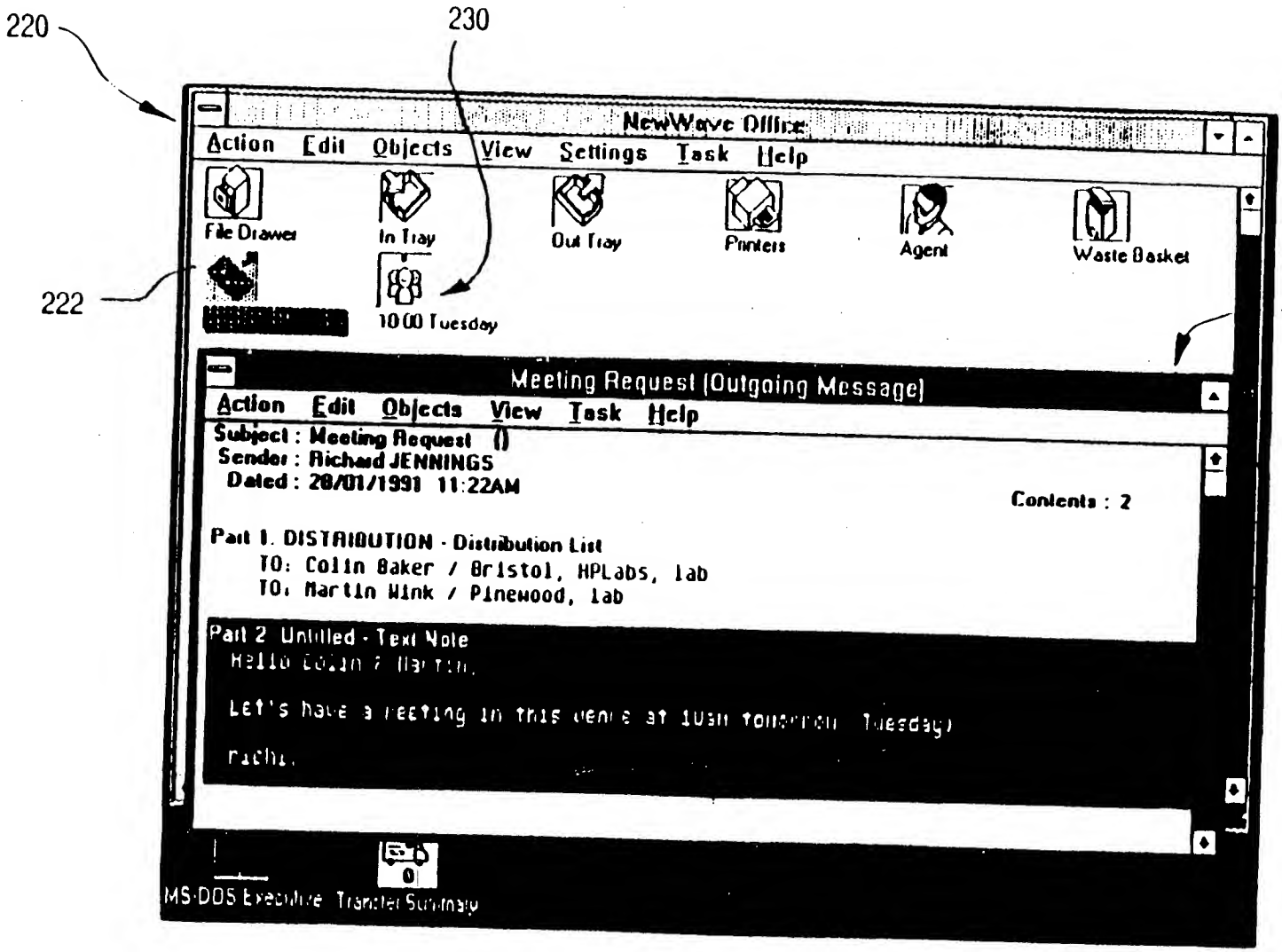


FIG 31

EP 0 497 022 A1

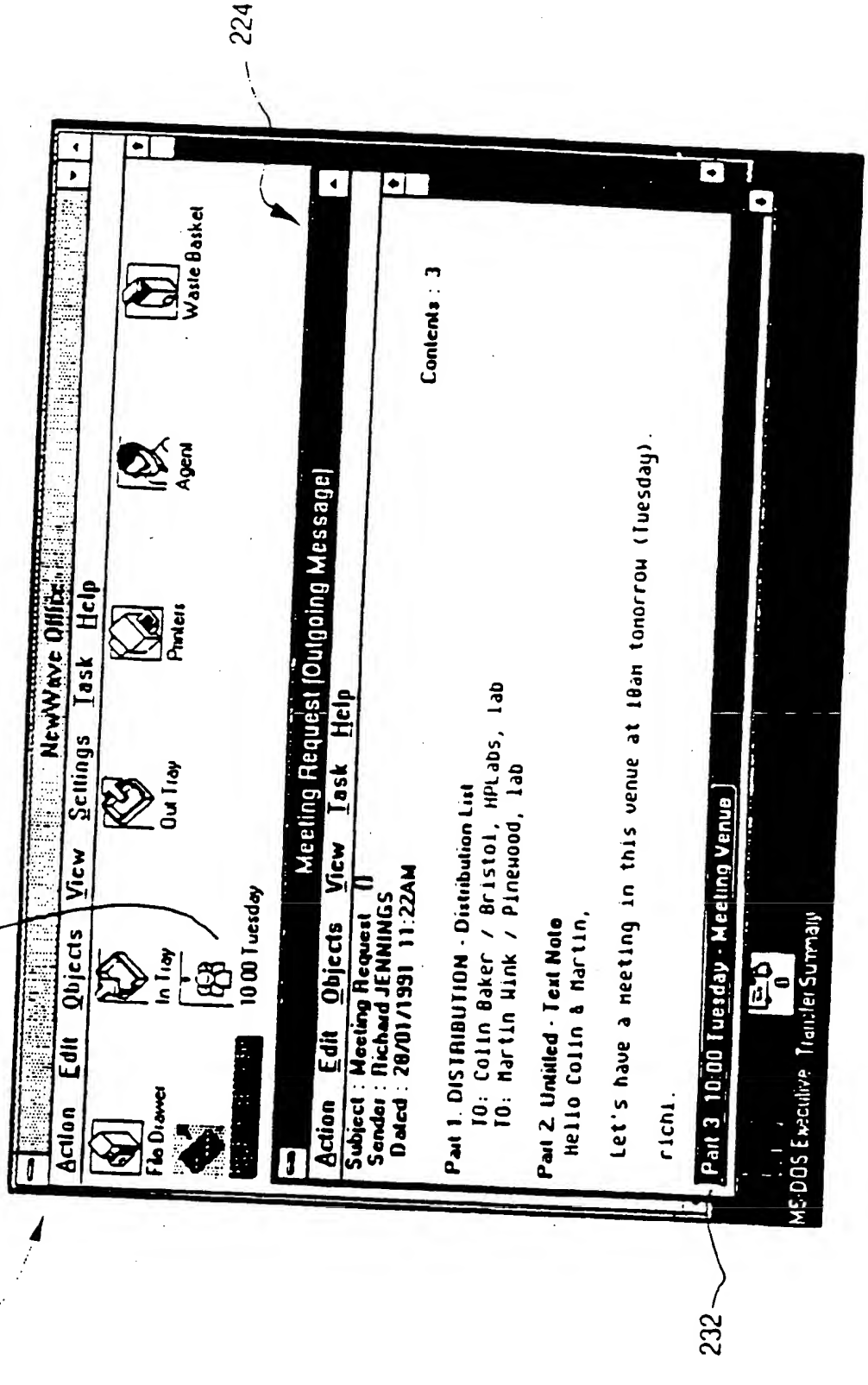

220
230
224
232
NewWave Office
Action Edit Objects View Settings Task Help
File Drawer
In Tray
Out Tray
Printers
Agent
Waste Basket
1000 Tuesday
Meeting Request (Outgoing Message)
Action Edit Objects View Task Help
Subject : Meeting Request
Sender : Richard JENNINGS
Dated : 28/01/1991 11:22AM
Contents : 3
Part 1. DISTRIBUTION - Distribution List
TO: Colin Baker / Bristol, HPLabs, Lab
TO: Martin Wink / Pinewood, Lab
Part 2 Untitled - Text Note
Hello Colin & Martin,
Let's have a meeting in this venue at 10am tomorrow (Tuesday).
richi.
Part 3 10:00 Tuesday - Meeting Venue
MS-DOS Executive
Trainer Summary

FIG 32

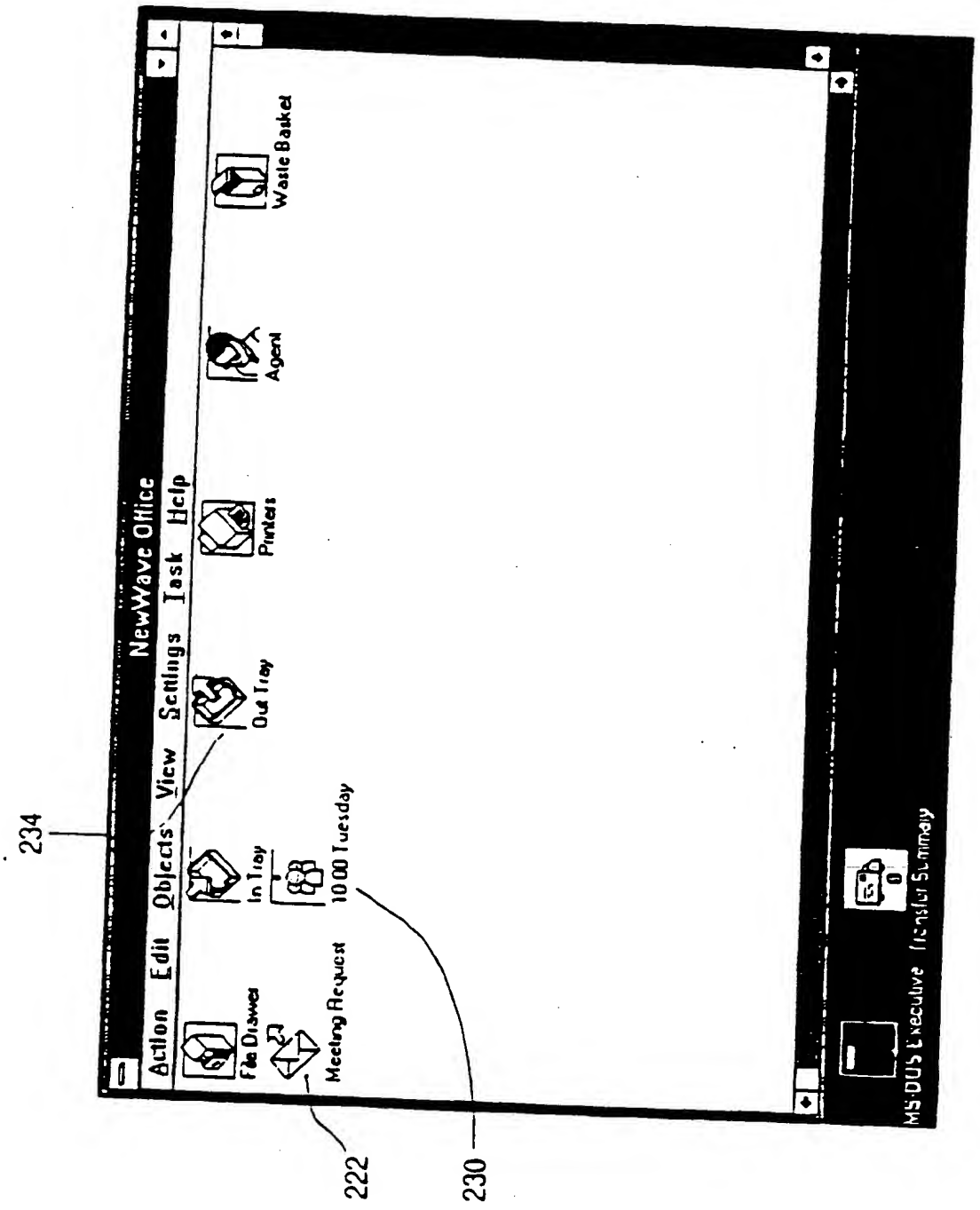

234
222
230
NewWave Office
Action Edit Objects View Settings Task Help
File Drawer
In Tray
Out Tray
Printer
Agent
Waste Basket
Meeting Request
10:00 Tuesday

FIG 33

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 91 30 0772

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X	EP-A-408 812 (HEWLETT PACKARD) * abstract * * page 3, line 29 - line 51 *	1-3	G06F15/40 G06F9/44 H04M3/56
Y		4,5	H04N7/14
Y	COMPUTER, vol. 18, no. 10, October 1985, SILVER SPRING, MARYLAND, US pages 33 - 45; SUNIL SARIN ET AL.: 'Computer-Based Real-Time Conferencing Systems ' * page 33, column 1, line 1 - page 38, column 1, line 44 * * page 39, column 3, line 31 - page 42, column 1, line 25 * * page 43, column 1, line 33 - page 44, column 1, line 25 *	4,5	

TECHNICAL FIELDS SEARCHED (Int. Cl.5)
G06F H04N H04M

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	18 SEPTEMBER 1991	KINGMA Y.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)